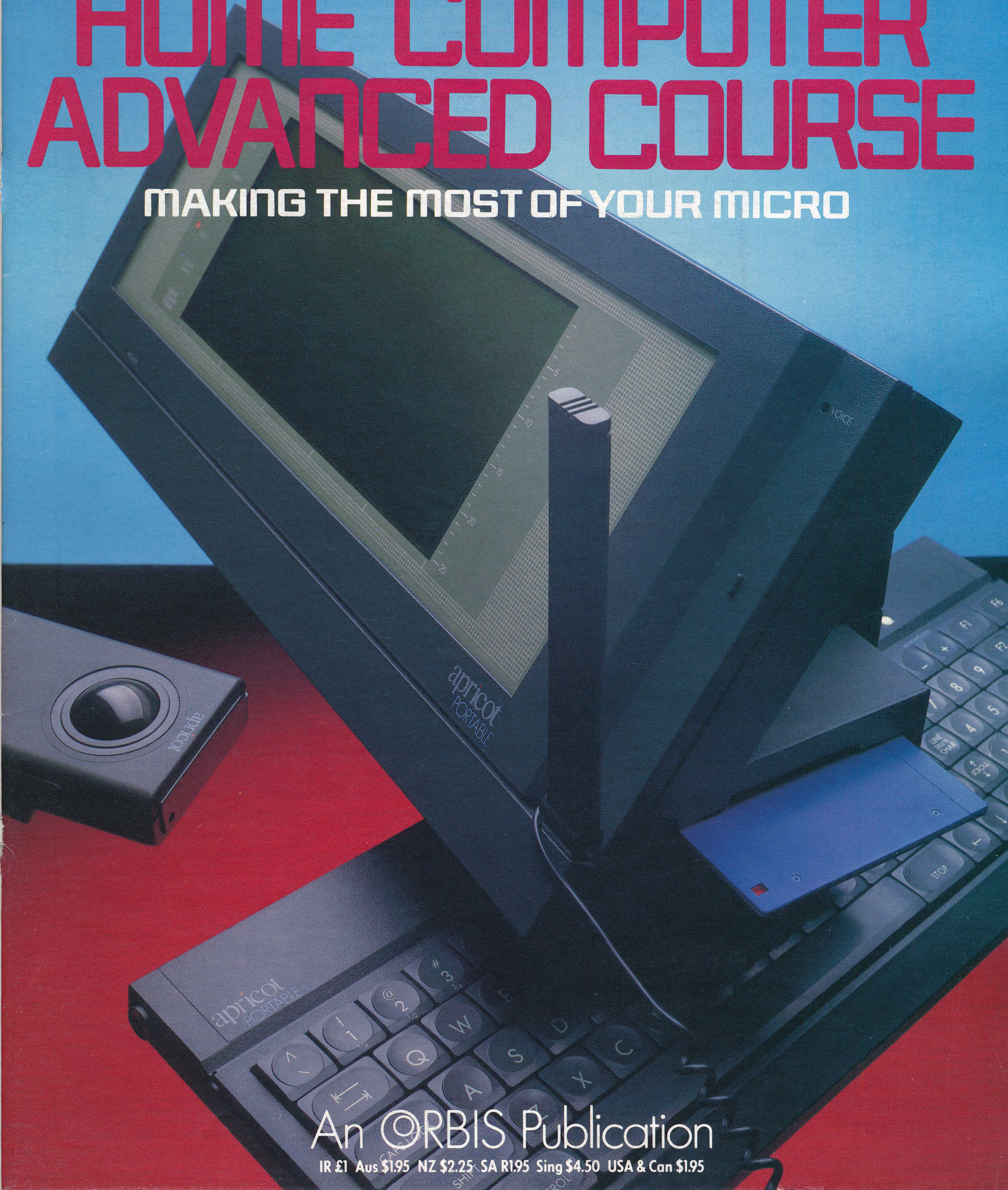
ISSN 0265-2919
80p 51

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO



An ORBIS Publication
IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

Next Week

- The CX5M from Yamaha is an MSX computer that is sold as a musical instrument. We examine this hybrid machine.
- "The Word" Processor is a software package on 7 disks that contains all the text of the King James Version of the Bible. We take a look at this remarkable program.
- Continuing our series on computers and education, we look at the problems involved in teaching young children using micros.

QUIZ

1) What problem could arise if several Apricot Portables are working in close proximity?
2) What are the three main stages of running a new PASCAL program?
3) How is the Calendar option used in the Microsoft Project program?
4)What is the most widely used application of magnetic cards?

Answers To Last Week's Quiz

1) Output devices from the Spectrum are signalled using the IORQ (Input/Output ReQuest) line.
2) 35.
3) The parts of a transistor are called the collector, the base and the emitter.
4) The custom-built video and audio chips are called the 'Nick' and 'Dave' chips, respectively.

Managing Editor Mike Wesley; **Editor** Stephen Cooke; **Art Editor** Claudia Zeff; **Production Editor** Bobby Pickering; **Technical Editor** Steve Colwill; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Staff Writer** Stephen Malone; **Sub Editor** Jonathan Kaye; **Contributors** Geoff Bains, Nick Walsh, Joe Pritchard, Steve Malone, Karl Dallas, Anthony Ginn, Steve Colwill; **Software Consultants** Pilot Software City; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Maurice Geller; **Production Manager** Peter Taylor-Medhurst; **Subscription Manager** Christine Allen; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 14 Rathbone Place, London W1P 1DE; **© APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Heanor Gate Printing Ltd, Derby**

HOW TO OBTAIN ISSUES AND BINDERS FOR THE HOME COMPUTER ADVANCED COURSE – Issues can be obtained by placing an order with your newsagent or direct from our subscription department. If you have any difficulty obtaining any back issues from your newsagent, please write to us stating the issue(s) required and enclosing a cheque for the cover price of the issue(s). **AUSTRALIA** – please write to: Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G, Melbourne, Victoria 3001. **MALTA, NEW ZEALAND & SOUTH AFRICA** – Back numbers are available at cover price from your newsagent. In case of difficulty, write to the address given for binders.
UK/EIRE – Price: 80p/IR£1. Subscription: 6 months: £23.92. 1 Year: £47.84. Binder: please send £3.95 per binder, or take advantage of our special offer in early issues. **EUROPE** – Price: 80p. Subscription: 6 months air: £43.68. Surface: £34.84. 1 year air: £87.36. Surface: £69.68. Binder: £5.00. Airmail: £8.25. **MALTA** – Obtain binders from your newsagent or Miller (Malta) Ltd, MA Vassalli Street, Valetta, Malta. Price: £3.95. **MIDDLE EAST** – Price: 80p. Subscription: 6 months air: £45.76. Surface: £34.84. 1 year air: £91.52. Surface: £69.68. Binder: £5.00. Airmail: £8.25. **AMERICAS/ASIA/AFRICA** – Price: US/CAN$1.95/80p. Subscription: 6 months air: £54.08. Surface: £34.84. 1 year air: £108.16. Surface: £69.68. Binder: £5.00. Airmail: £9.50. **SOUTH AFRICA** – Price: SA R1.95. Obtain binders from any branch of Central News Agency or Intermag, PO Box 57394, Springfield 2137. **SINGAPORE** – Price: Sing $4.50. Obtain binders from MPH Distributors, 601 Sims Drive, 03-07-21, Singapore 1438. **AUSTRALASIA/FAR EAST** – Price: 80p. Subscription: 6 months air: £58.24. Surface: £34.84. 1 year air £116.48. Surface: £69.68. Binder: £5.00. Airmail: £9.75. **AUSTRALIA** – Price: Aus$1.95. Obtain binders from First Post Pty Ltd, 23 Chandos Street, St Leonards, NSW 2065. **NEW ZEALAND** – Price: NZ$2.25. Obtain binders from your newsagent or Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington.
ADDRESS FOR BINDERS AND BACK ISSUES – Orbis Publishing Limited, Orbis House, Bedfordbury, London WC2 4BT. Telephone 01-379 5211. Cheques/postal orders should be made payable to Orbis Publishing Limited. Binder prices include postage and packing and prices are in stirling. Back issues are sold at the cover price, and we do not charge carriage in the UK.
NOTE – Binders and back issues are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK and Australian markets only. Binders and Issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.
ADDRESS FOR SUBSCRIPTIONS – Orbis Publishing Limited, Hurst Farm, Baydon Road, Lambourn Woodlands, Newbury Berks, RG16 7TW. Telephone: 0488-72666. All cheques/postal orders should be made payable to Orbis Publishing Limited, Postage and packaging is included in subscription rates, and prices are given in sterling.

COVER PHOTOGRAPHY BY CHRIS STEVENS

COMPUTER TUTOR

TONY LODGE

In this introduction to a series of articles analysing the impact of computers on education we outline the theories, experiments and government-funded schemes that sought to mechanise the learning process in the days before the advent of the ubiquitous micro.

'In the long term we expect the effect of computer related systems of information to be as far reaching on the education system as was the development of the printed book.' These words, from a government report published in 1978, have more recently been echoed by Professor Tom Stonier, a leading authority on new technology. Both he and others, including the US educationist Seymour Papert, believe that schools as we know them will disappear within our lifetime.

This rather radical view is a natural consequence of the level of commitment these individuals have to computerised education. In reality, however, change comes very slowly and is driven by the people in education rather than the machines at their disposal. Nevertheless, there are many people, both within the educational system and outside it, who feel that, used properly, computers can make a significant impact on the learning process.

The use of computers in the classroom can be traced back to various educational projects and experiments conducted over the last 20 years, although some of the fundamental principles involved have been around for much longer. Simple teaching machines had been constructed in the 1920s, although they did not become popular until the 1950s, when B F Skinner created a primitive feedback learning machine. Skinner, one of the most influential educational psychologists of the 1950s and 60s, developed his

The Changing Chalkface
In the modern day classroom, a computer can, with appropriate software, easily replace the blackboard, overhead projector, TV, cassette deck, and library. Where direct replacement is not possible — in the 'painting corner' of a primary classroom, for example — computers can still usefully supplement traditional teaching methods. More importantly, the teacher is freed from onerous time-consuming tasks (marking texts, and so on) and given more time to interact with each student on an individual basis

ideas over a 20-year period while experimenting with rats and pigeons. One of his more unusual projects involved a homing device for a missile consisting of three pigeons in its nose cone, thus anticipating Cruise by 20 years. Skinner had conditioned the birds to peck at a point on a map by rewarding them with food. The project, incidentally, was rejected by the Pentagon, leaving 'a loftful of curiously useless equipment and a few dozen pigeons with a strange interest in a feature of the New Jersey coast'.

Skinner's ideas on learning were better received, however. He advocated a system of 'linear programming' (nothing to do with computer programming) that involved presenting information to a student one small piece at a time and positively reinforcing a correct response. In effect, Skinner applied the same principle to students as he had to his 'pigeon pilots'.

LINEAR PROGRAMMING

Two important concepts came out of linear programming — 'feedback' and 'individualisation'. Feedback refers to the immediate response given to the student, telling him whether the answer given is right or wrong. Unfortunately, Skinner's learning machines, which relied on cranks and push buttons to operate, could accept only specific answers. Individualisation refers to the direct nature of the method, which enables the student to work at his own pace rather than that dictated by his classmates or teacher.

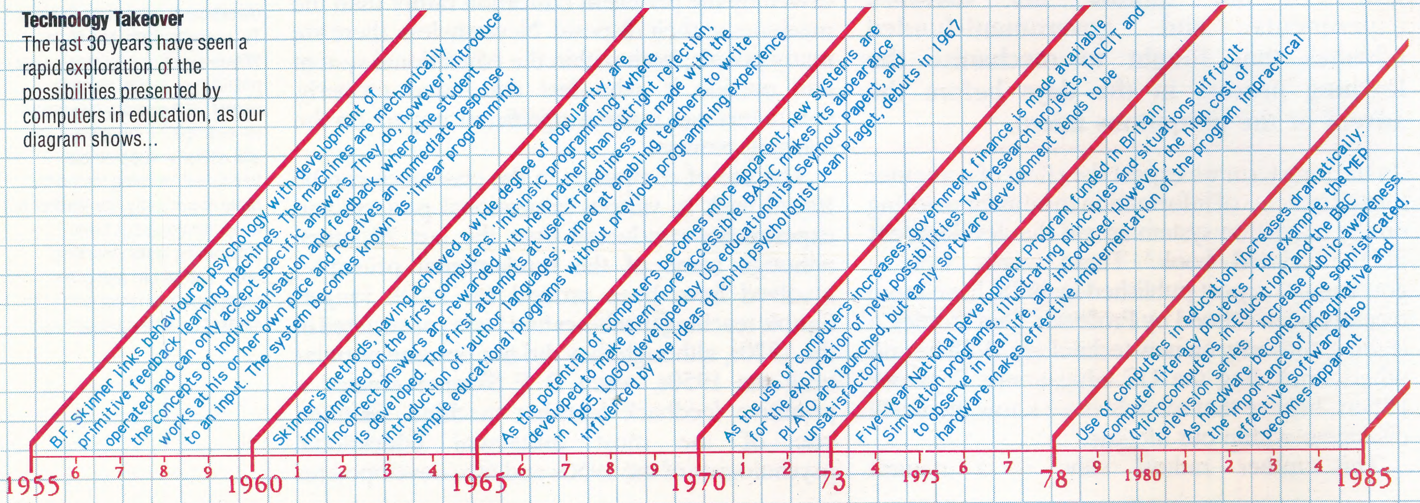
Computers became common while Skinner's teaching machines were at the height of their popularity, and linear programming systems were soon implemented on them. Despite the vast limitations of this method, we still frequently encounter programs that use it. 'Sorry, have another try' or 'Well done, you've got it right' flash up on the screen, reinforcing responses to a boring stream of questions.

One development of linear programming was the 'branching program'. Instead of merely having another go if the incorrect answer was given, the student received help and was then re-tested. The approach seems logical, but 'intrinsic programming' (as it became known) greatly upset Skinner and his followers. However, for the student, feedback was greater and more individually tailored than before and it corrected misunderstandings. 'Author languages' were devised to help teachers write this type of program. They were an early attempt at user-friendliness, supposedly enabling a teacher to write an educational program without having to learn computer programming.

Both linear and intrinsic programs have serious drawbacks. They look upon the student as an empty vessel to be filled with knowledge, and see learning as acquiring facts rather than experience. They leave no room for self-expression, imagination or creativity, and encourage a single 'correct' response. These systems, which give us the term 'programmed learning', have unfairly earned a bad reputation for all computer-assisted learning.

Despite the restricting influence of the early teaching machines, computers were soon recognised as offering entirely new possibilities in education. Much of the early work done in this area has been of lasting importance — and not only for school teachers. In 1965, a project at Dartmouth College in the USA set out to devise a teaching language 'as near to English as possible' to enable scientists, engineers, and students to write their own programs. Because of the hardware limitations of the time, when a 16 Kbyte machine cost about £40,000 and filled a living room, the language had to be designed to fill as little memory space as possible. The result was the Beginner's All-purpose Symbolic Instruction Code, or BASIC, and it has been with us ever since.

A couple of years after the creation of BASIC, a team of computer scientists and educationalists at the Massachusetts Institute of Technology devised a programming language 'designed to be easy for the programmer rather than the computer'. The objective was to enable very young children to control a computer, a revolutionary idea at the time, as previous projects had all been targeted at undergraduates. The head of the project was Seymour Papert, who had studied with the child psychologist Jean Piaget. Piaget's ideas about



Technology Takeover
The last 30 years have seen a rapid exploration of the possibilities presented by computers in education, as our diagram shows...

education were radically different from those of Skinner. In contrast to the latter's mechanistic approach, Papert believed in creating an environment in which children could learn by discovery. He proposed that: 'Rather than have the computer program the child, allow the child to program the computer'. The result of his work was LOGO, a language that has recently become available on micros and is proving popular in primary schools. The advent of BASIC and LOGO made direct programming of computers accessible to students even at primary level.

Despite all the energy being invested in computer-assisted learning techniques, many people still had doubts about the benefits of using computers in education. In an effort to resolve some of this uncertainty, the National Science Foundation of America decided in 1970 to invest $10 million over five years to investigate the subject. Two projects were set up — TICCIT and PLATO.

TICCIT AND PLATO

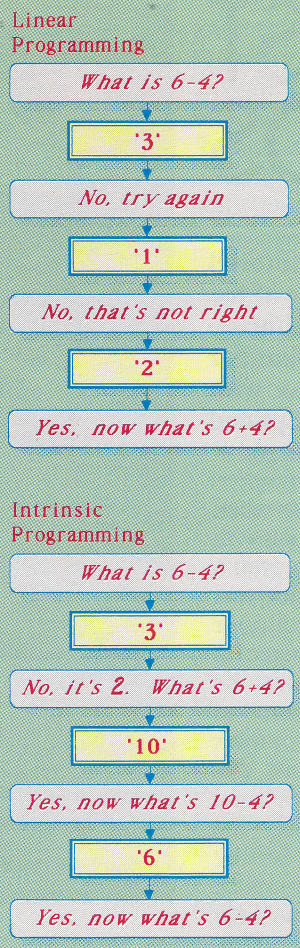
The aim of TICCIT (Time-shared Interactive Computer Controlled Information Television) was to 'provide better instruction at less cost than traditional instruction in community colleges'. A company was commissioned to produce hardware and software for education and devised a keyboard for the student that included special 'learner control' buttons, a television display for text, graphics and videos, and a loudspeaker for audio messages. These were configured in systems of 128 terminals, controlled by two minicomputers. There were problems with the software and many students did not complete the course, but those who did achieved higher test scores than those who studied by conventional methods. The TICCIT system has remained in use in the two pilot colleges, but has not been taken up elsewhere.

PLATO (Programmed Logic for Automatic Teaching Operation) consists of a network of nearly 1,000 terminals connected to a mainframe computer in Illinois. The terminals can communicate with each other and access a library of lessons held in the mainframe. PLATO display units have transparent panels to allow colour slides to be projected onto computer graphics. A touch panel enables students to communicate with the program by touching the screen. Teachers can write their own programs in an author language called TUTOR. As with TICCIT, software preparation took more time and effort than was envisaged. The results of the students using PLATO were no better or worse than those of students using conventional methods, although the students using PLATO found the system more user-friendly than those working with TICCIT.

In 1973 the British government sponsored a £2.5 million, five-year National Development Program in computer-assisted learning. Most of the money went on developing software in the form of simulations, the computer being used to create situations difficult or impossible to observe in real life. One program, for example, simulated the concentration of dye in the blood stream at different time intervals and others illustrated chemical reactions. However, most of the software was unimaginative and the high cost of computers in the 1970s made their use impractical.

Any educational system is influenced by politics, economics, and the demands of society. Computers in education not only have to contend with these factors, but also with the pressures of technological change. To that must be added the conservatism of those who see computers as representing simply an extension of the undesirable 'programmed learning' techniques discussed earlier.

Meanwhile, as in all areas of computing, the dominating trends in educational computing today result from the rapidly falling price of memory. There is now more computing power in one London primary school than in all the British universities of 20 years ago. One effect of this, according to some critics, has been the concentration on hardware provisions at the expense of software development. However, politicians now seem more prepared to recognise software as a national resource of vital importance and, as the price of hardware continues to fall, we can expect to see radical changes in our educational systems over the next ten years.



A Helping Hand

Using linear programming systems, the student must enter the correct response to a question before proceeding further. The system rejects incorrect responses and then re-presents the original question without offering the student any further assistance.

With intrinsic programming systems, however, an incorrect entry by the student will not simply result in the same problem being encountered again immediately. Instead, the system will respond with an error message and a series of different, but related, questions. Once these have been answered correctly, the student will be re-tested on the original problem

IAN McKINNELL

FIRST WORDS

In the second instalment of our tutorial series on PASCAL we run through some of the fundamentals of its syntax and vocabulary. Programmers are weaned off the familiar but less efficient characteristics of BASIC with examples that demonstrate the disciplined structure and economy of the language.

Popular Compilation
A full professional implementation of PASCAL can often cost more than the price of a home computer, but recently there has been a wave of reasonably priced compilers for popular home micros. We show a selection of packages now available

The problem normally encountered first when using a compiled language is learning to cope with the multi-stage process required to obtain even a small executable program. First, the source text has to be entered via some form of text editor or word processor. Then, having saved the source on tape or disk, the language compiler must be loaded and instructed to compile the source to some form of machine code (often with a complex string of command line options). Lastly, this 'object' file must be re-located and linked with any run-time library routines needed. Possibly the program can be loaded and run without further effort, but should the compiler use a 'pseudo-code' or intermediate code, then a run-time interpreter must be used to execute the program.

If all this sounds a little too formidable, take comfort from the fact that all the PASCAL packages available for home computers avoid these problems to a large extent. At the very least during program development your source text, compiler and object program can be co-resident in memory. PASCAL's efficiency and small size makes this possible, and the resulting system is often no more complicated in use than a resident BASIC package. It's only when we start writing programs of any significant size that we need to resort to the more complicated processes described above.

Each system will have its own set of commands to control the editor and compiler, and you must refer to the relevant documentation for help with these. Very often, a simple E for edit, C for compile and R for run will be all that is required. What concerns us is the correct syntax to be entered for every program, no matter how trivial or complex. Fortunately, PASCAL is so well standardised that there will be little or no need for the many programming 'flavours' that we have given in the past to deal with the different dialects of BASIC (although later in the course we will have to deal with some minor exceptions). So let's look at our first complete PASCAL program:

```
Program First (output);
Const
     Message = 'Pascal Programming';
Begin
     write (Message)
End.
```

Before we study this example in depth, try entering, compiling and running it. If you get any complaints from the compiler, read the error message carefully and see if you can spot what's wrong. Every symbol must be presented exactly as shown. Take special care with the semi-colon at the end of the first and third lines and the full stop at the end of the program. When you have successfully executed this program you will see the following message displayed on the screen:

```
Pascal Programming
```

The program may be trivial but it demonstrates the overall form that every PASCAL module (program, procedure or function) will have. There are three separate parts:

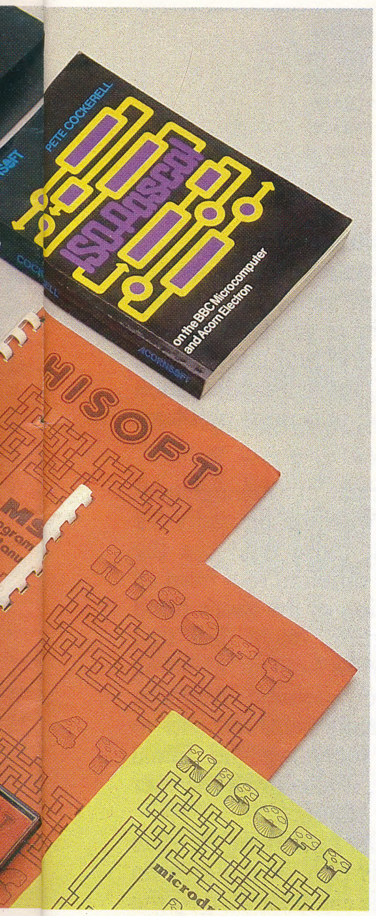
1. The heading, in this case a program heading.
2. Declarations and definitions, here a single constant definition.
3. The 'body', which contains all the executable statements.

The syntax requirements of PASCAL, at least for the fundamentals of the language, may best be defined by means of 'syntax diagrams'. These are like the road map of a one-way system. A legal route through the diagram proceeds from top left to bottom right, and every box that we pass through is either a 'syntactic entity' (that is, it

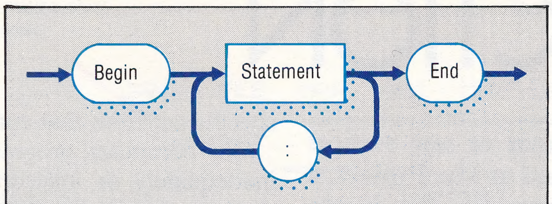
Package: HISOFT PASCAL
For: Spectrum, Amstrad, MSX machines
Price: Spectrum (£25) Amstrad and MSX (£29.95)
Description: A slightly non-standard implementation, but a bargain nonetheless. The package comes with its own editor and a turtle graphics library

Package: Acorn ISO PASCAL
For: BBC Model B, Electron
Price: £69
Description: Excellent value. There are two compilers: one in ROM, with a semi-intelligent editor and memory-to-memory compilation; the other a disk-based system for second (6502) processor owners

Package: Oxford PASCAL
For: Commodore 64
Price: £49.95 (disk) £19.95 (cassette)
Description: Another bargain. Limbic Systems, which produces the package, is expected to release versions for the BBC Micro in February, 1985 and the Spectrum in April, 1985



represents itself) contained in a round-edged box, or another item described elsewhere by a separate syntax diagram, indicated by the rectangular box in which it is enclosed.



LIZ DIXON

Syntax Diagram Of 'Begin — Statement'

Referring first to the overall diagram of a program, it can be seen that the words Begin and End are defined to be part of PASCAL'S vocabulary, and require no further diagrams to elaborate their meaning. In fact, PASCAL recognises only 35 words as having a fixed meaning, and for the sake of completeness we list all of them here. Our first program makes use of only four of them — Program, Const (short for constant), Begin and End. The word following Program is an 'identifier' that identifies the program's name and can be any legal identifier you choose.

If we start with a letter and use only letters or digits there are an almost infinitely large number of names we could use. Predictably, however, it is impossible to use a reserved word. For example:

```
Name
PASCAL
ProgramOne
N
XYZ123
Address12
FloraMacDonald
AVeryLongIdentifierIndeed
```

are all legal, whereas:

```
Prog-1
TEN%
Lumber.Jack
and
Time$Square
IDon'tKnow
1001Dalmatians
It figures
```

are illegal, either because they contain symbols which are not alphanumeric, start with a number or (in the case of and) duplicate one of PASCAL'S reserved words. The last example is illegal because a space is used to separate the words, each component word (It and figures) being quite legal. In PASCAL, as in English, there is no difference in meaning between lower-case letters and capitals, though some non-standard versions may insist on reserved words being in upper case.

Apart from spaces and the end of a line, there is another item of PASCAL syntax that can be used as a separator — a comment. This may appear anywhere in the text except, of course, in the middle of words. Comments are delimited by 'brace' (curly bracket) symbols.

Let's be a little more adventurous and enter something that looks like a real PASCAL program:

```
Program ProgramTwo (input, output);
  {Gives the square of a number}
Const
      Prompt ='Enter a number:';
Var
          number : integer;
Begin
  WriteLn;
  WriteLn;
  write (Prompt);
  read (number);
  WriteLn (number, ' squared is ', number * number)
End.
```

Notice that we now include the identifier input in the heading. Input and output are required in PASCAL, and they identify the external files with which the program will communicate. Normally on a micro they will be the keyboard and VDU screen respectively. By including input, we can now read from the keyboard using the standard PASCAL procedure read. As with write, any 'parameters' must be listed in brackets.

The memory used to store these parameters is reserved by the Var declaration, in this case for a single whole number. Unlike BASIC, which can usually distinguish only between numbers and strings (by using a dollar sign after the identifier), PASCAL has an almost unlimited range of data types available. It therefore becomes important to let the compiler know how much memory to reserve for storage of each data item. You must always declare every single variable in a Var declaration.

The cursor will remain positioned immediately after the prompt as if we had used a BASIC PRINT statement with a final semi-colon. This is not a bug but exactly what we asked for by using PASCAL'S built-in write procedure. Whenever a new line is required, we must use the alternative procedure WriteLn. The Ln is a contraction of the word Line, and it is helpful to use a capital W and L to indicate the start of each of the component words. A simple WriteLn statement without parameters will merely create a new line.

PASCAL enables us to differentiate between data that varies and items that will remain constant throughout the execution of the program. Notice, therefore, that the Const definition uses an equals sign, whereas the Var declaration uses a colon.

PASCAL's 35 Reserved Words

					In	Not	Procedure	Set	Until
And	Case	Do	End	Function	Label	Of	Program	Then	Var
Array	Const	Downto	File	Goto	Mod	Or	Record	To	While
Begin	Div	Else	For	If	Nil	Packed	Repeat	Type	With

FORWARD PROJECTION

In the previous instalment of our vertical software series, we looked at MacProject, a powerful graphics program for the Apple Macintosh that applies the principles of critical path analysis to long-term project management. Here, we look at a similar application for MS-DOS machines, such as the IBM and its compatibles.

Although Microsoft's Project package does not have the sort of graphics capabilities of MacProject — or of rival Digital Research's Graphics Environment Manager program — it makes the best possible use of the IBM system's limited graphics potential. For instance, combining equals signs or dashes with greater than symbols can result in some useful arrows:

======>   ------>

The first can be shown in bold to signify critical paths, and the second in a lightweight print to show those activities that are not critical. The salient factors that have to be considered, however, are more involved than this. They are:

- How flexible is the program, and how responsive is it to the sort of changes likely to become necessary as the project develops?
- How clearly does it display the relationships between activities, and does this assist in efficient management of the project?
- Is it possible to use the program as an aid in the initial planning, or must users fully develop their ideas first?

To answer the last question first, the user really needs to sort out the project on paper before keying in the first activity. Indeed, the manual states quite unequivocally:

'Before you enter activities into Microsoft Project, you need the following information about each activity:

What it is (description)
How long it takes (duration)
What has to be done beforehand
(predecessors)
When it starts (start date).'

To emphasise the importance of such preparation, the manual offers a 10-page appendix, 'Defining Tasks by a Work Breakdown Structure', which states, amongst other things:

'Successful project management starts at the beginning of a project. First, you must state clearly the project's goals, purpose, constraints, and specifications. Then you must define the set of specific activities that constitute the project.

'No computer program can compensate for inadequately or inaccurately defined activities. Activity definitions may have implications for resource scheduling, allocation and control, and for budget and cost control. Inaccurate definition of activities is likely to result in serious problems once the project begins.'

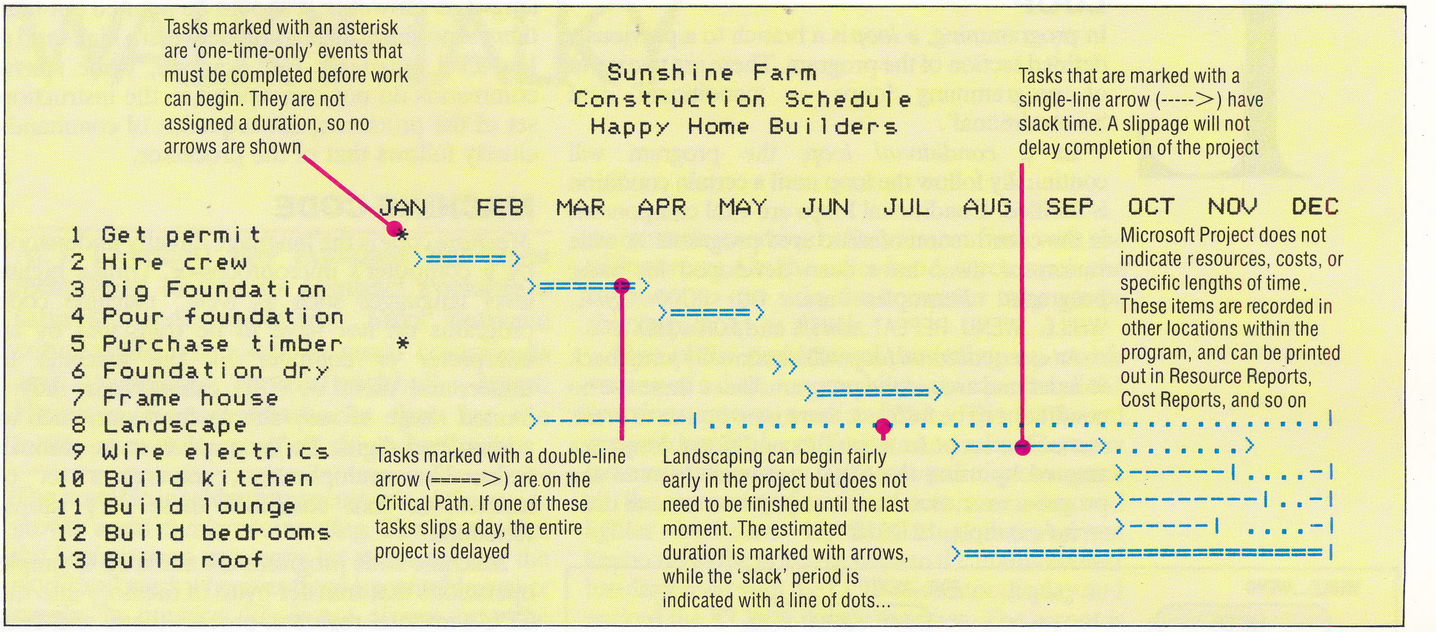
Few of those experienced in management techniques would quarrel with the necessity for adequate preparation. But those who see the computer as a qualitatively different tool from all others, acting as an extension of the user's mind rather than of the fingertips, might feel that software's task is precisely to help in avoiding inadequately or inaccurately defined activities. But Microsoft Project certainly helps the user to get the priorities right. Including the initial recourse to pen and paper, it also gives the user plenty of opportunities for changes.

When the computer is first turned on, the date and time, which are set as part of the sign-on sequence, will determine the calendar used by the program — but the calendar can be changed quite easily, even after several activities have been typed in. If the time allocated to any activity is altered, as soon as the change has been input, the path diagram is recalculated changing some activity arrows to bold ===> (critical), others to light ------> (non-critical). This not only makes the relationship between activities very easy to observe, but it also instantly highlights a change that may not otherwise be obvious to the user. It would appear from this that Microsoft have applied their experience in producing Multiplan, one of the most successful spreadsheets since the genre was pioneered by Visicalc in 1979. In fact, using Microsoft Project is rather like using a spreadsheet, and anyone familiar with using one will find themselves at home with Project almost immediately.

USING MICROSOFT PROJECT

After a brief display of the Microsoft logo, the activities screen is displayed (it is blank if no filename has been specified), with numbers from one to 19 down the left of each row and the timescale along the top, beginning with today's date if that was input initially. This may be changed with the OPTIONS item on the menu along the bottom of the screen — again in the style of Multiplan — which also includes:

BLANK CALENDAR DELETE EDIT GOTO HELP INSERT MOVE PRINT QUIT RESOURCE SORT TRANSFER XTERNAL



Construction Project
This chart represents a plan for the construction of a house. Dates appear across the top of the chart, and each task appears, in the order of operation, along the left-hand edge. The duration of each task is represented with arrows constructed from keyboard characters (< > . = -)

Beneath the legend 'Select option or type command letter', the current option in use is displayed (initially ACTIVITY) and at the bottom right corner, the name of the Project in use, automatically given the title TEMP until the user names it otherwise.

The timescale may be calibrated in days, weeks or months, and this may be changed at any time to assist in viewing the entire project at a glance. With a daily timescale, only the dates between 1 January and 2 March, say, can be seen, whereas a monthly timescale permits two years and three months to be seen at once. The right and left cursor control keys are used to scroll the ACTIVITY screen sideways, in the usual spreadsheet manner. Using the CALENDAR option, the user programs in holidays and other non-working days first; Saturdays and Sundays are normally scheduled as non-working. This can be changed, however, but it has to be done for each individual day. Later months can be displayed with the <PAGE DOWN> key, and previous months can be returned to with <PAGE UP>.

Like most date options with IBM and compatible machines, care must be taken to utilise the American order of MM/DD/YY (for instance, 01/07/85 is 7 January 1985, not 1 July 1985). Once the calendar has been adjusted, it can be SAVEd using the TRANSFER option in usual Multiplan style. The program adds the suffix .CAL to distinguish it from activity files (suffix .ACT) and resource files (.RES).

The user then begins to type in activities, the time taken for each, and states the preceding activities it depends on — its 'predecessors' — with multiple dependencies separated by commas. The column for Activities Titles is limited to 15 characters, and this may not be widened. However, if more than 15 characters are typed in, they will be held in memory and included in any hard copy print-out. As each activity is typed in — with its duration, predecessor(s) and start date — the relationships between the various activities are immediately displayed, with slack time (the amount an activity can be delayed without delaying the total project) represented as dots. At this point, resources are allocated also. As soon as any resource has been entered, it is possible to call up the Resource Table to monitor the results.

When the data has been entered, the complete schedule can be printed out on a page-by-page basis. But if a graphics card has been fitted and the appropriate printer connected (IBM PC's graphics printer or Epson's MX-80, MX-100, FX-80 or FX-100), it is possible to have the whole project plan turned sideways and printed along the length of the paper. This is useful, as it means that a full plan can be printed out, without needing to cut-and-paste printouts. It is also possible to obtain detailed reports on each of the activities, showing the earliest and latest start and finish times, the resources, predecessors, and slack times, and a table of activities for the entire project displaying the same information in less detail.

In practice, the lack of sophisticated graphics in the Microsoft Project is less of a disadvantage than might be initially thought, and though the excellent manual rightly emphasises the necessity of adequate pre-planning, it is still possible to adjust the information after second or third thoughts — or unexpected delays. The program will be particularly friendly to anyone experienced with spreadsheets, especially Multiplan, because its method of operation is very similar. It is faster in operation than the Macintosh MacProject, but then it is almost three times the price.

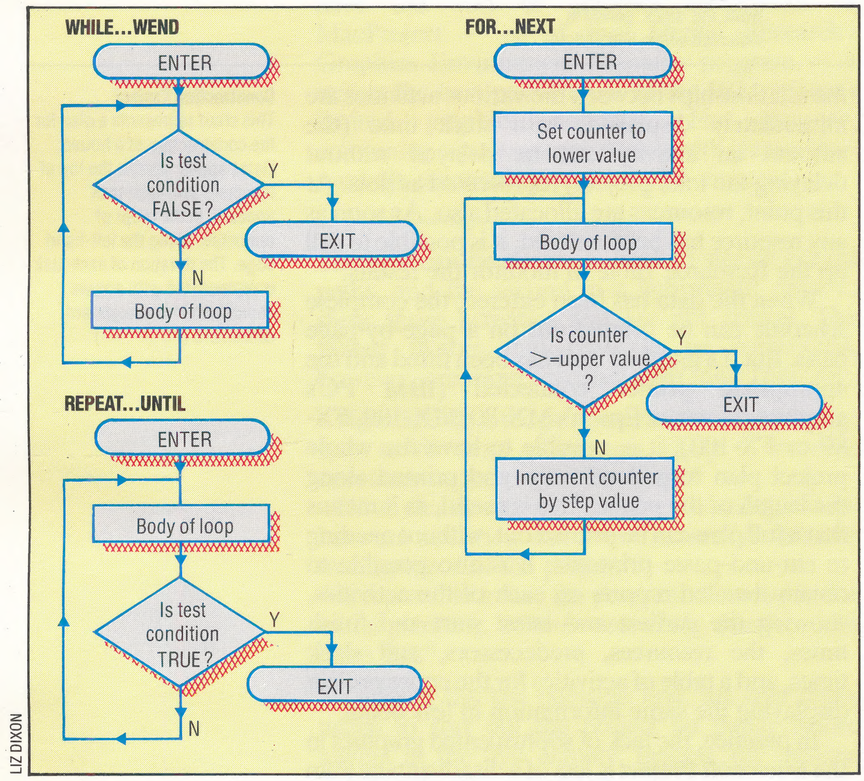
Microsoft Project: For MS-DOS machines
Price: £270.25 (inc VAT)
Distributors: Microsoft Ltd, Piper House, Hatch Lane, Windsor, Berkshire
Authors: MAS, Seattle, USA
Format: Disk

LOOP

In programming, a *loop* is a branch to a previously defined section of the program. There are two types of programming loops — 'conditional' and 'unconditional'.

In a *conditional loop,* the program will continually follow the loop until a certain condition is fulfilled. Conditional loops are vital components in the construction of structured programs. A wide variety of these have been developed for BASIC programs. Examples are: DO LOOP...UNTIL, WHILE...WEND, REPEAT...UNTIL and FOR ...NEXT.

An *unconditional loop* will continually jump back to a defined area of the program. Since there are no conditions to be fulfilled, there is no option to break out of the loop. In BASIC, unconditional loops are created by using the GOTO statement to send the program execution back to a previous area of code — for example, 10 GOTO 10.



Going In Circles
The diagram shows three common conditional loop structures. WHILE...WEND and REPEAT...UNTIL have certain similarities, but the loop termination test is performed in different places in each structure

LOW-LEVEL LANGUAGE

Programming languages are generally compromises between two imperatives. On the one hand — for maximum efficiency — the language should reflect the actual workings of the processor; and on the other hand, it should be as intelligible as possible. A language whose commands correspond closely to the instruction set of the microprocessor is called a *low-level language.* Because the language is so closely linked with specific operations within the computer, a low- level language requires little, if any, interpretation. This produces a corresponding acceleration in the computer's functioning.

Examples of low-level languages are hexadecimal machine code and Assembly language. However, it should be pointed out that there is no sharp distinction between a high- and a low-level language. For example, while FORTH commands do not correspond to the instruction set of the processor, the sequence of commands closely follows that of the processor.

MACHINE CODE

Machine code is the language directly understood by a computer's microprocessor. Unlike higher level languages such as BASIC, machine code programs do not have to be translated by an interpreter or compiler for the machine to understand them. A CPU can perform only a limited range of very simple functions, such as adding two digits. To accomplish more difficult tasks, like multiplication, requires a set of instructions that combines these very simple functions.

Machine code programs are made up of simple operations that transfer bytes of memory into the CPU's internal registers, process them, and then return them to a specified location in memory. Instead of using words like PRINT or GOTO, machine code programs manipulate individual bits or bytes of memory — using the simple logic functions such as AND, OR, and NOT, and performing simple binary arithmetic. These operations are not intuitive, and they require a detailed knowledge of the computer's memory locations and internal registers. This makes machine code programming a daunting proposition for many people, and problem-solving a long and arduous task.

The advantage of programming in machine code over BASIC and other high-level languages is the speed with which operations can be carried out — direct access to the CPU makes this possible. Graphics routines in arcade games nearly always rely on machine code to a certain extent precisely because such speed is required. Still, BASIC and other languages do some things very well, so it is common to find many programs written largely in BASIC that refer to machine code routines only for specific functions.

MACHINE INDEPENDENT

Most software is designed to run on a specific machine that has a set of carefully defined characteristics. Because of the lack of standardisation in the microcomputer industry, programs written to take advantage of, for example, the graphics characteristics of the Sinclair Spectrum, have to be completely rewritten if they are to run on the Commodore 64. Some programs, however, are written in such a way as to be independent of the unique qualities of a particular machine, and can therefore run equally well on a variety of computers. Software of this type is *machine independent,* and is also referred to as 'portable' software. This portability is generally possible only between machines that have a common operating system, such as CP/M or PC/DOS.

WALKY TALKY

ACT (Apricot) has exploited the latest technology to give its compact Portable a futuristic feel: cable-free links between components, response to vocal commands and versatile mouse/trackball input. However, it has drawbacks that for the business user may offset these facilities.

The portable machine represents one of the major growth areas in microcomputing. The idea is that the businessman can carry his micro around the world and work wherever there is a suitable power supply. In practice this is not always how the machine is used. Often a portable computer will be kept at home and work from a compatible machine is taken home in the evening to be completed on it.

ACT (Apricot) is one of the latest of a long line of computer manufacturers to produce a portable machine that is compatible with its best-selling line of desk-top computers.

The Apricot Portable is supplied within a sturdy plastic carry case about the size and shape of an attache-case but with a bulge at the bottom to accommodate the computer screen. The case is opened by pressing two clips on either side of the handle, although on the review machine this proved rather difficult.

HARDWARE ANALYSIS

Inside, the keyboard and computer are held on either side of the case by straps. Also provided within the case are two manuals, a microphone and a mouse/trackball. Unusually, no cable is provided to attach the keyboard and the mouse to the computer. This is because the data is transmitted via an infra-red beam. Data is transmitted and received by two small bulbs on the back edge of the keyboard and on the front of the mouse. Signals from these devices are picked up by a large focusing bulb on the computer. Thus, it is possible for the user to operate the keyboard on his lap and position the computer several feet away on a table.

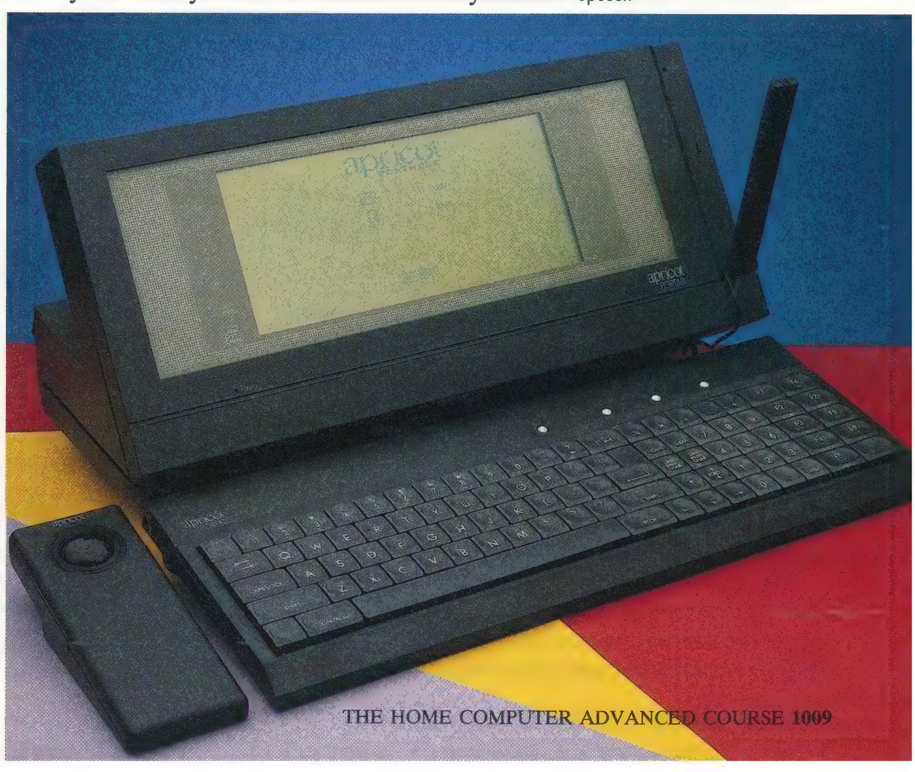
There are two disadvantages to this infra-red system. First, if several Apricot Portables are running in the same room the infra-red signals from different machines may interfere with each other. To avoid this, Apricot has made a light tube available. This is a fibre optic cable that connects the keyboard to the computer in the conventional way. The second problem occurs when using the mouse. If the user positions the keyboard directly in front of the computer, it will often block the signals from the mouse. This is because the computer's receiving bulb is almost flush with the table surface. On the other hand, if you move the keyboard to one side, the computer may not be able to pick up its signals.

One of the problems of producing a portable version of a desk-top machine is to squeeze all the functions into a smaller space. The keyboard presents particular difficulties and manufacturers have used all kinds of ingenious techniques to pack all the necessary functions onto a smaller and lighter keyboard. In translating the Apricot keyboard to the portable version the manufacturer has dispensed with the LCD calculator display and packed the 87 keys tightly together. The layout is identical to that of the desktop Apricot, except for the function keys, which are on the extreme right instead of above the main layout.

Partly to compensate for the loss of the LCD display, which in the desk top version is also used for single keypress commands, Apricot has added an extra function key, making a total of nine, and a key to display the time and date. The QWERTY keys on the left-hand side of the keyboard are flanked on either side by control keys, the cursor keys being at bottom right. Between the typewriter keys and the function keys is a calculator keypad, with the number keys in a lighter shade of grey than the rest of the keyboard.

Although the keyboard appears extremely elegant, the keys are perhaps positioned a little too close together for comfort. Instead of the conventional indented keys, the Portable has a QL-style flush keyboard. This makes the keyboard

Moveable Feast
The Apricot Portable uses some of the very latest technology. The machine is equipped with an 80 by 25 text screen, which allows a wide variety of MS-DOS applications to be displayed. Also shown here is the keyboard and mouse/trackball: these are not connected to the computer in the conventional sense, but transmit signals via an infra-red beam. On the right-hand side of the computer is the microphone, which provides the computer with the facility to recognise speech



CHRIS STEVENS

more compact, as the keys do not protrude from the main body of the keyboard, but many typists will find the keyboard design awkward to use as it is difficult for the fingers to differentiate between the keys. The compactness of the arrangement emphasises this problem: there is no gap between the typewriter, the calculator and the function keys. However, the layout has its compensations. Because of the design, the Shift, Stop and Caps Lock are twice the size of those on the Apricot, making them easier to locate.

Above the keyboard and in place of the calculator display are four buttons to perform extra functions. There is a Reset button, which reboots the disk currently in the disk drive. The second button allows the user to alter the repeat rate of the keys. The third calls a routine that resets the time held in the computer, and the fourth locks the keyboard. This last is useful when you are using the mouse or speech input and do not want any spurious information input by the accidental press of a key.

The built-in LCD screen is fitted onto the main body of the computer. Its clarity is poorer than on comparable machines even though the light olive background is brighter than most. To the right of the screen is a microphone, which is connected by a thin wire that plugs into a microjack socket on the left of the computer.

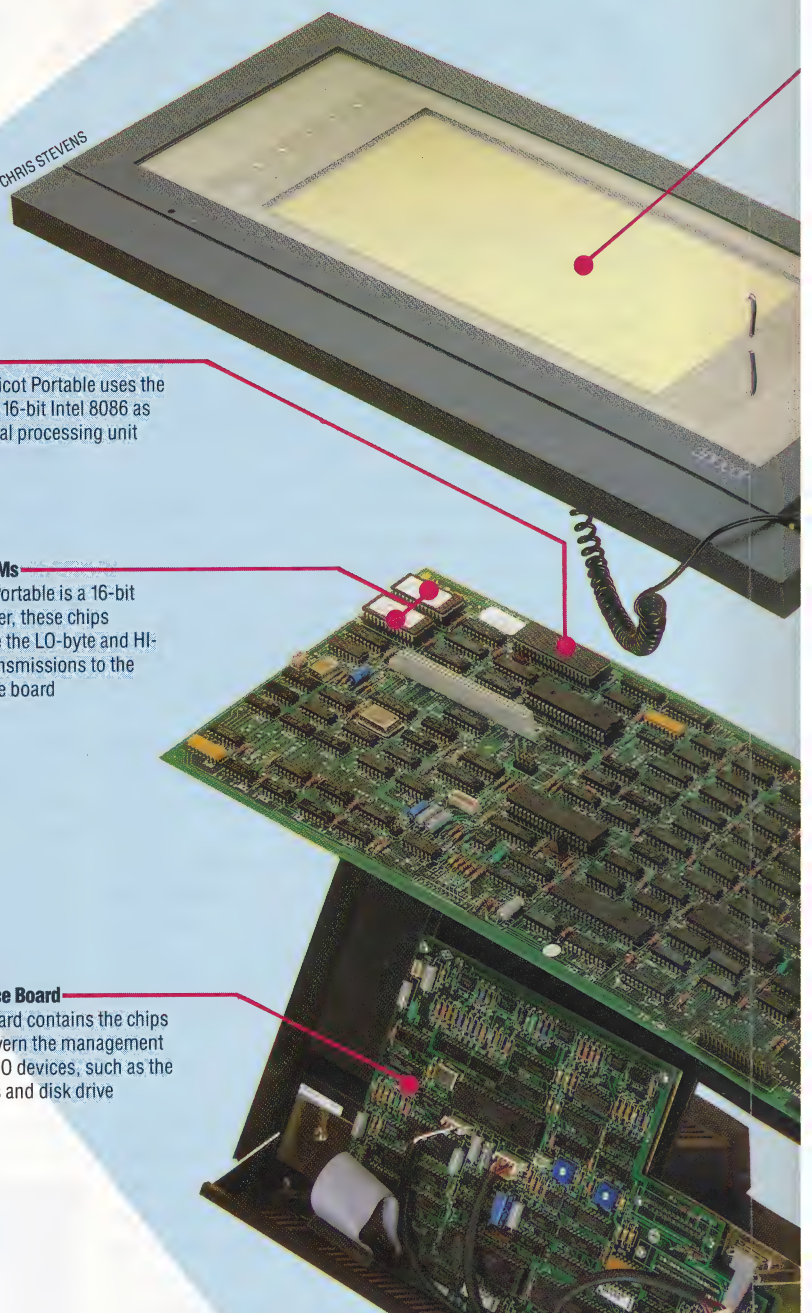
SPEECH RECOGNITION

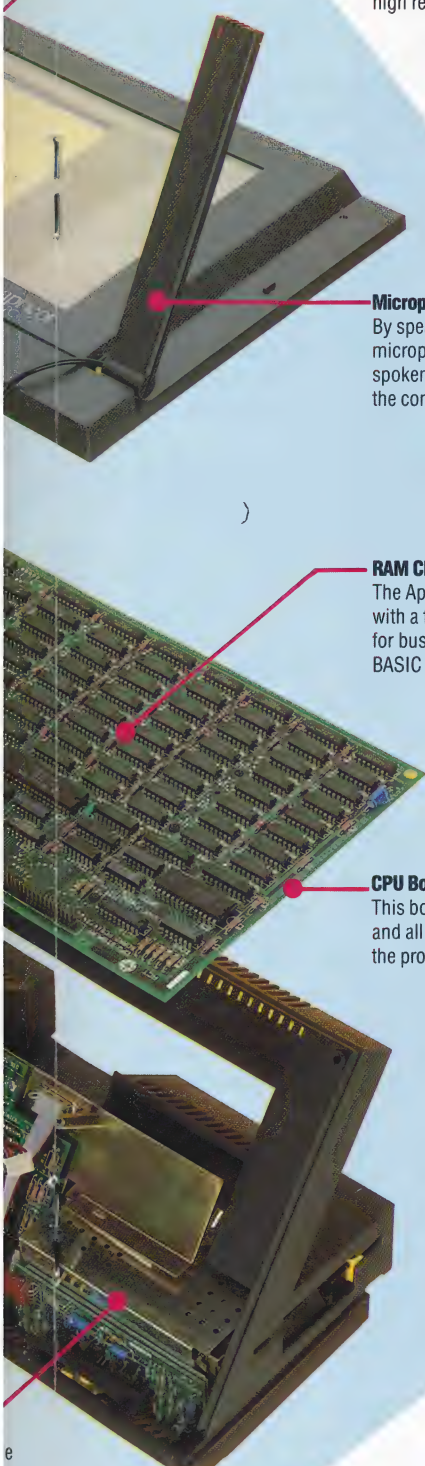
Perhaps the most interesting feature of the Portable is its speech recognition program, which recognises vocal commands, passes them to the input and then executes them in the normal way. Voice Driven Application mode accepts vocabularies of up to 63 words for any applications program, such as WordStar or SuperCalc. A voice training program, it will ask the user whether the voice to be used will be male, female, or a child's, and whether or not there will be much background noise. After speaking a command several times, the computer will accept and compare several variations. However, it still misinterprets words and often ignores them altogether. For this reason, DELETE and FORMAT should not be commanded vocally.

In common with the rest of the Apricot range, the Portable uses Sony 3in disks to hold the applications programs. The single disk drive is positioned on the right-hand side of the computer. On the rear of the machine are three interfaces which are covered by a protective plastic casing. These are a Centronics parallel slot for printers, an RS232 socket for an external modem or other serial device and an Atari-standard joystick port. This joystick slot is provided not to enable games playing but to fit a mouse or trackball in circumstances where the infra-red mouse is inappropriate.

The battery-driven infra-red mouse provided with the Portable is cleverly designed so that it can either be used as a mouse that can be slid around a desktop or table or, reversed, it can be held in the hand or kept in a fixed position and used as a trackball. The mouse actually works better as a trackball, for the user will often find that it has been pushed too far across the desk top and moved out of the range of the infra-red receiver on the computer.

The Apricot Portable is certainly an interesting machine to use. ACT has gone to great pains to include the latest technology in its design. However, many business users have no intrinsic interest in leading-edge technology; what they want is a reliable machine that runs their applications programs easily and comfortably. By this criterion the Apricot Portable scores less well. The difficulty of reading the screen and the fallibility of the input devices may result in business users turning to more tried and tested technology.





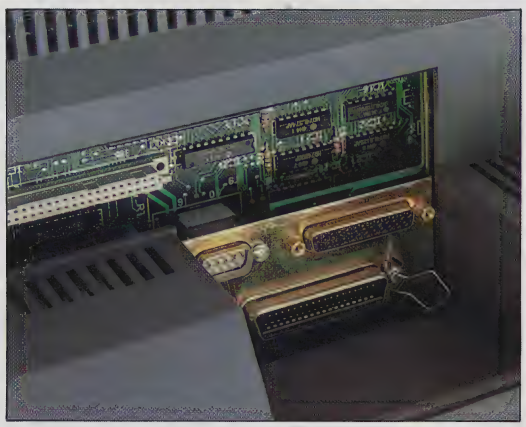
LCD Screen
This screen gives a full 80 by 25 text display. However, it is also possible to produce graphics on the screen of a high resolution

Microphone
By speaking into the microphone, the user can give spoken commands directly to the computer

RAM Chips
The Apricot Portable is fitted with a full 256 Kbytes of RAM for business applications and BASIC programs

CPU Board
This board contains the CPU and all the other chips used by the processor

e
sity
ricot



Peripheral Interfaces
At the back of the computer, shielded beneath a plastic casing, are the peripheral interfaces. At the bottom is the Centronics parallel port, and above this, to the right, is the RS232 socket (which allows the computer to be connected to an external modem). On the left is a port through which a conventional mouse or trackball can be attached

Infra-Red Trackball
By moving the ball around, the cursor arrow travels in a corresponding direction on the screen. When the cursor is pointing at the chosen icon, pressing one of the buttons on the side accesses the chosen application. The device can be turned upside down so that the ball rolls along the desk-top, thus allowing the peripheral to act as a mouse

CHRIS STEVENS

Apricot Software
Provided with the Apricot Portable are a number of applications packages that allow the user to run various different programs. These applications are held on 3in Sony disks. Among the packages provided are SuperCalc (a spreadsheet), SuperPlanner (a card index and calendar), and SuperWriter (a word processor)

APRICOT PORTABLE

PRICE
£2064.25 (including VAT)

DIMENSIONS
450×335×200 mm

CPU
Intel 8086

MEMORY
32K ROM and 256K RAM, of which 211K is free for applications programs

SCREEN
Electrofluorescent LCD screen producing a text resolution of 80×25 characters and a graphics resolution of 640×256 pixels. If the computer is plugged into an external monitor, up to 16 colours can be displayed

INTERFACES
RS232 serial, Centronics parallel, and a mouse connection. There is also a 'light pipe' connection for using the infra-red-based keyboard and a jack socket for the microphone

DISK DRIVES
The portable contains a 720K double-sided Sony 3 in disk drive

OPERATING SYSTEMS
MS-DOS, CP/M-86, and Concurrent CP/M

KEYBOARD
IBM-compatible keyboard containing 87 keys with ten function keys. There are also four function buttons

DOCUMENTATION
A Starter Pack manual, giving details on how to set up the computer and how to use the operating system and speech recognition facilities. The Applications Pack manual provides a tutorial guide to the applications of SuperCalc, SuperPlanner and SuperWriter

STRENGTHS
The ACT Apricot Portable is certainly one of the most advanced machines on the market today. The technology developed for use on the computer will doubtless win the machine many admirers

WEAKNESSES
Because the technology used in the machine is so advanced it is not yet infallible. The LCD screen can be extremely difficult to read

BRIDGING THE GAP

In our previous Workshop instalment, we showed how an interface could be designed for the Spectrum that would mimic the user port. Here, we construct the interface, which will enable the Spectrum user to control the Workshop robot.

The edge connector designed for the Spectrum should have the fifth position from left (looking into the connector) blanked out. So, our first task is to insert a blanking plug into the edge connector that will plug into the Spectrum's expansion port. This plug ensures that the finished interface can only be inserted into the socket properly orientated. Maplin do not supply a suitable plug, but one can be easily made.

After cutting the veroboard into the shape shown, one of the corners can be used as a blanking plug. First, strip off the copper track and jam the square of stripped board edge-on into the fifth position of the edge connector — using a little superglue for good measure — then cut off both pin connections. Now make thorough track cuts on the veroboard, using either a 'spot face cutter' (available from Maplin; part number FL25O) or a 5mm ($\frac{1}{4}$in) twist drill bit. Make sure that each cut breaks the whole width of the track, or this could mean that besides your interface not working, the Spectrum itself could be damaged.

Having made the track breaks, we can start to assemble the components onto the board, beginning with the wire links. The links should be cut with length to spare and then inserted loosely into the board. Solder one end and pull the loose end taut before soldering the other. Finally, the extra lengths should be cut off close to the board with a pair of wire cutters. When all the links have been soldered in place, wire in the insulated flying lead, using a short length of one strand of the ribbon cable.

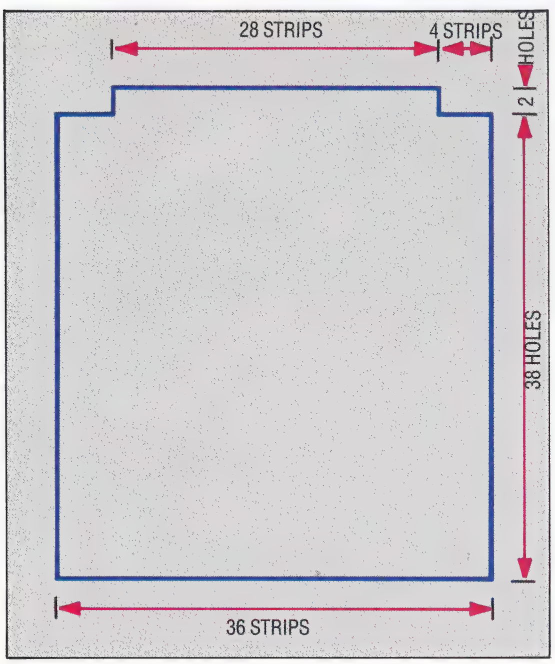
After affixing all the links, the edge connector can now be soldered into place. But because the edge connector system is really designed for double-sided boards (as used in the Spectrum itself) and not the single-sided veroboard, it is a tricky operation. First, bend one row of pins on the edge connector as shown. These will form the bottom row of pins when the edge connector is positioned on the board. Now, solder 3cm ($1\frac{1}{4}$in) lengths of tinned wire onto each of the pins in the top row. Insert the bent pins into the row of holes nearest the board edge and bend each of the lengths of wire through 90°, inserting them into the other row of holes (marked on the diagram). Now put a little superglue on the edge of the board between the two cut-out corners and press the edge connector home. Solder each pin carefully to the tracks of the stripboard. Before continuing, check the copper side of the board and clear any solder that is bridging across the tracks, using a sharp knife.

Now solder the rest of the components into their places on the board. The three DIL sockets should be soldered first, and then the four K-ohm resistors. Finally, insert the three chips into position, making sure that you insert them the right way round. The notches at one end of the chips should all be pointing in the same direction — to the right when the board is held with the edge connector uppermost.

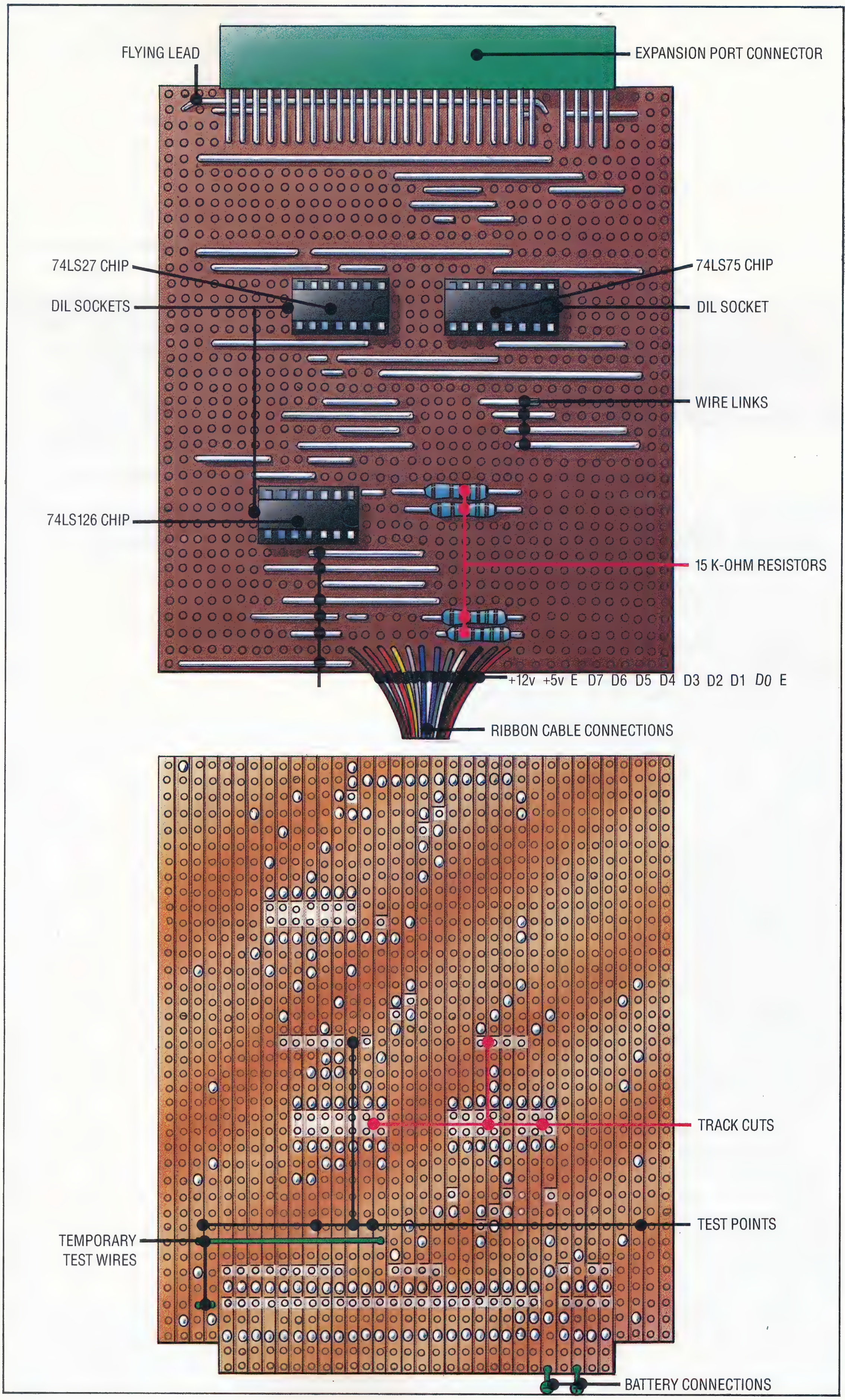
The interface is now complete. However,

Parts List

No	Item	Maplin Number
1	74LS27	YF18U
1	74LS75	YF32K
1	74LS126	YF50E
4	15K-ohm, 0.3 watt	M15K
1	Reel 20 swg tinned wire	BL13P
2	14-way DIL socket	BL18U
1	16-way DIL socket	BL19V
5m	20-way ribbon cable	XR07H
1	2×28 way 0.1 in edge connector	FG23A
1	15-way D connector	BK58N
1	15-way D connector cover	BK60Q
1	36 strip × 50 hole veroboard	FL09K



Cutting Remarks
The Spectrum interface components mount onto a piece of veroboard cut to this shape. Use one of the small corners removed as a blanking plug in the expansion port connector. The extra lip on one end of the board will support this connector when it is soldered in place



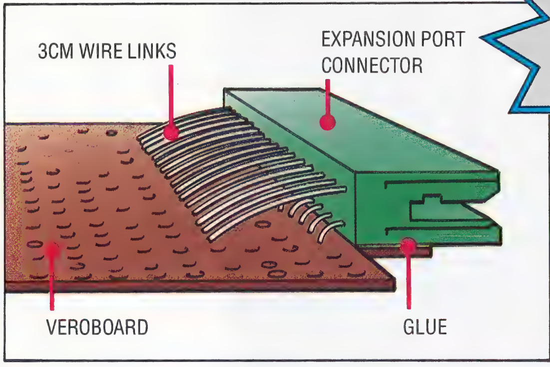
Board Layout

Solder the wire links and resistors in place on the component side of the board first, then fit the expansion port connector and DIL sockets. Ensure that all three chips are placed in the DIL sockets correctly orientated, so that all the notches point the same way, as shown. Check all your work very carefully against the diagram before attempting to plug the interface board into your Spectrum.

On the copper underside, make sure that all track cuts break the track completely, using a meter to test for continuity. The links marked in green are temporary connections for electronic testing of the board and should be removed after successful completion of the test. As always, check your work thoroughly for mistakes

before you can plug it into your Spectrum, it is important to thoroughly test it. Since the Spectrum's expansion port allows us unrestricted access to the address and data buses, processor pins, and the 12v and 9v supplies, it is important that we do not inadvertently connect these voltage supplies to the delicate electronics that run at TTL levels. A visual test should be made first. Go over the entire board carefully. Check that there are no bridges of solder between tracks at any point, that all the components and wire links are in the right place according to the diagrams and that the chips are positioned the right way round.

In the next instalment of Workshop, we shall be wiring the ribbon cable that connects the robot to the interface board, and looking at Spectrum versions of some of the previously written robot programs.



Sideways Look

To allow us to use a standard edge connector for the interface board, we must make slight modifications to the pins. Having removed the fifth position pins, bend over the lower set of pins at right-angles and solder 3cm lengths of tinned wire to the upper set of pins. Push the pins through the relevant holes in the board, checking that each pin locates correctly. The bottom row of pins should locate in the third row of holes and the upper row of pins locate in the sixth row of holes. Glue the connector in place before soldering the pins to the board

WARNING!

The Spectrum's expansion port allows us access to the delicate electronics inside the computer. Also present at the expansion port are voltage outputs that, if inadvertently routed down the data or address bus lines, would cause major damage to your computer. Check ALL connections very carefully before plugging the interface into the port and switching on

Testing The Board

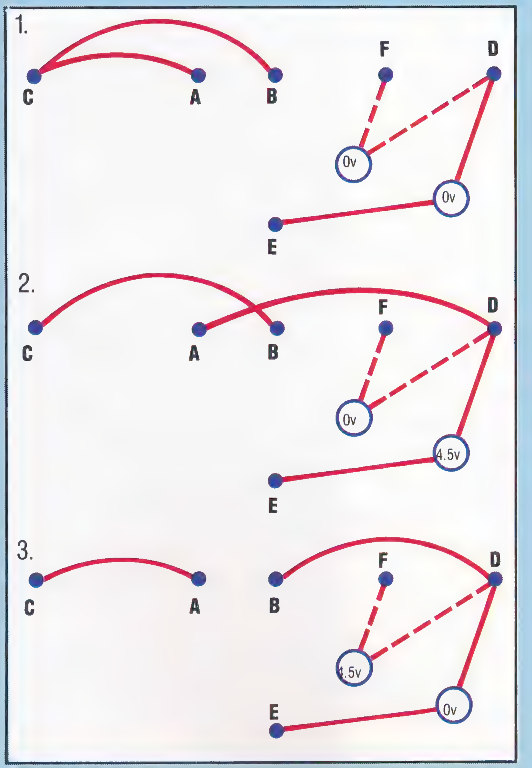
Before plugging the interface board into the Spectrum's expansion port, we can make a series of electrical checks to ensure that the decoding logic is working correctly. To do this we need a multimeter, a 4·5v battery, and several short lengths of covered wire. First, make the two connections shown in green on the copper side of the circuit board and connect the battery terminals to the correct points on the board. Use two lengths of covered wire to connect point A to point C and point B to point C. As always, check your work carefully and clear any excess solder from between the tracks. Having made these connections we can now test the voltage levels at points E and F. With the negative meter probe on point D, touch the positive probe first on point E and then on point F. Both these readings should be within one volt of 0v.

Unsolder the wire between A and C at C and reconnect the loose end at point D. With the negative probe of your meter still on point D, take new readings at E and F. E should now read 4·5v and F should still be 0v.

Disconnect the wire from A to D at D, and the wire from B to C at C. Swop the loose wire ends over and remake the connections so that B connects with D and A connects with C. Again take readings at E and F. F should now read 4·5v and E should read 0v.

If all these tests yield the correct readings at points E and F then remove from the board the battery connections, the two green wire links, and the flying leads between A and C, and B and D. If, however, any of these readings are not as stated, then check the construction of the entire board again — very carefully. Under no circumstances should the interface board be plugged into the Spectrum until the test readings are correct. The various test connections and correct readings are given in the table.

Connections	Readings
A-C, B-C	E=0v F=0v
A-D, B-C	E=4.5v F=0v
A-C, B-D	E=0v F=4.5v



This diagram shows the test connections to be made between the points A, B, C, and D, and the correct readings that should be obtained at points E and F in each of the three test stages

FILED AWAY

We conclude our look at the BBC Micro's filing system ROM routines. There is a variety of operating system calls that deal with data sent to and accessed from storage media. We begin here by considering the OSFIND call, a routine that allows us to perform 'byte access'.

OSFIND is called at address &FFCE and is used to open a file for what is called 'byte access'. This is simply the same as opening a file in BASIC using OPENOUT, OPENIN or OPENUP. It also enables us to close files, in a similar way to the CLOSE# operation in BASIC. The A register specifies the operation to be performed and the X and Y registers point to the file name in memory.

- *A=0:* This indicates that a file is to be closed. Here, the Y register holds the 'number' that the file was given when it was opened. Setting the file number to 0 will result in all currently opened files being closed.
- *A=&40:* This is equivalent to the BASIC OPENIN command. A file is opened for the input of data (bytes) only.
- *A=&80:* This is the same as a BASIC OPENOUT. A file is opened for output only.
- *A=&C0:* This is equivalent to the BASIC OPENUP. Here, a file is open for both input and output. It is particularly applicable to any file system that can support random access files.

When we're using OSFIND to open a file, however, we don't always succeed. For instance, there might be a write protect label on the disk. If the file cannot be opened, then A will be set to zero on return from OSFIND. If the file is opened successfully, the file 'number' is returned in the A register. This should be saved for future use.

The next few filing system ROM calls that we'll look at all have one thing in common; they all use the file number that is generated by OSFIND.

THE OSARGS CALL

This routine is called at address &FFDA and performs a variety of jobs. It has one peculiarity — its parameter block, which is only four bytes long, is stored in the zero page. The X register holds the address of the first byte of the parameter block, Y holds the file number and the A register specifies the operation to be carried out. There is a special case of OSARGS, when the Y register is set to zero. In this case, the routine returns information about the filing system in use rather than the file, or ensures that all open files on the filing system are up to date with regard to data that might still be in the buffers for the files. Let's look at these two routines first.

- *Y=0:A=0:* A call to OSARGS with these parameters and the X register pointing to a free area of the zero page will return filing system information. The data will be returned in the A register. This OSARGS call is the only one that will work on a tape filing system. The data returned in the A register by this call is interpreted according to the table in the margin.
- *Y=0:A=255:* The other call with Y set to zero, is when A holds the value 255. A call to OSARGS under these conditions will ensure that each file is updated from the buffers of the machine. If no files are open, then the updating cannot take place.
- *Y=File Number:* If the Y register contains the file number, then the operations performed depend on the contents of the A register (as shown in the following table). The 'sequential pointer' of an open file indicates to the operating system where the next byte will be written or read. It is therefore changed by the BASIC PTR# function, as well as by these routines. As we might expect, data is stored in the zero page parameter block with its LSB in address (X+0). When data is read from the filing system by OSARGS, it is stored in the parameter block for access later.

Contents of A Register	Description
0	Read the present position of the sequential pointer in the file indicated in the Y register
1	Write the value in the parameter block to the sequential pointer. Similar to the PTR# n=value instruction
2	Put the file length into the parameter block
255	Update the file (its number is given in the Y register) to the media

From our discussion of this call, it is clear why the OSARGS parameter block must be in the zero page — there's only the X register to hold the address.

OSBGET AND OSBPUT

These routines are used, respectively, to read single bytes from an open file and to write single bytes to an open file. OSBGET is called at address &FFD7, and is simple to use. The Y register is loaded with the number of the file – provided by OSFIND – and a byte is read in by calling &FFD7. The byte is read from the current position of the sequential pointer and is returned in A.

Contents of A Register	Description
0	No filing system
1	1200-baud tape
2	300-baud tape
3	ROM
4	DISK
5	Network
6	Telesoft

OSBPUT is the reverse of this operation; it writes a byte to the file at the current position of the sequential pointer within that file. To use OSBPUT, the byte to be written to the file is placed in A, while Y holds the file number. A call is then made to &FFD4 to execute the call. The short routine that follows shows how we might use OSBPUT to put a single byte in a file.

```
LDA   #&80 / prepare to open a file
LDX   #name MOD 256
LDY   #name DIV 256
JSR   OSFIND   / open the file
STA   &71      / store the file number
TAY            / also put it in Y register
LDA   #65      / the byte to be written
JSR   OSBPUT   / do it
LDA   #0
LDY   &71      / prepare to close file
JSR   OSFIND   / close the file
RTS            / finish
```

THE OSGBPB CALL

This is called at address &FFD1. It transfers groups of bytes between memory and open files. It can also provide us with information about the Disk Filing System, if it is in use. However, here we'll just consider its usefulness in data transfer. By the way, the call isn't implemented on the Tape Filing System – which isn't too surprising, because it's only of real use when we're using random access files from machine code.

The parameter block for the call is shown in the following table, and is pointed to by the contents of the X and Y registers, as usual. The A register specifies the operation to be performed.

OSGBPB Parameter Block

Address	Description
0	File number, as provided by OSFIND
1	LSB of address of data for read or write
2	Not specified
3	Not specified
4	MSB of address of data
5	LSB of number of bytes to be moved to or from the file
6	Not specified
7	Not specified
8	MSB of number of bytes to be moved
9	LSB of sequential pointer position to be used. (Thus we can select where in the file we write or read the data)
10	Not specified
11	Not specified
12	MSB of sequential pointer position

Again, if you're using a standard BBC Micro then the address is specified by the contents of bytes 1 and 4 in the parameter block.

There are four different types of data transfer operation available to us: two read and two write operations. One read and one write operation access the information in the file at the location specified by the sequential pointer entry in the above parameter block, whereas the other read and write operations ignore this information entirely and use the sequential pointer at whatever position it happens to be in the file. Thus A=1 writes bytes to the pointer specified in the block; A=2 writes bytes to the file ignoring the parameter block entry; A=3 reads data from the sequential pointer specified in the parameter block; and A=4 reads bytes but ignores the parameter block. The following program uses OSGBPB calls to write and read a simple data file:

```
10REM disk system only
20DIM  block 20
30DIM data 100
40$data="This is a test program!"
50 block!1=data:REM address of data to be
 written
60block!5=100:REM amount of data to be written
70block!9=0:REM sequential pointer=0
80 PRINT "Opening the file for output"
90Y%=OPENOUT("FILE"):REM open the file
100block?0=Y%         :REM file number
110X%=block MOD 256
120Y%=block DIV 256
130A%=1: REM prepare to write data
140 CALL &FFD1 :REM  do it
150CLOSE £(block?0)
160 PRINT "Close the file"
170 TIME =0: REPEAT: UNTIL TIME=400
180 PRINT "Opening the file for input"
190PRINT "Now read the data in!"
200$data=STRING$(100," "):REM clear data area
210Y%=OPENIN("FILE"):REM open the file up
220block?0=Y%        :REM file number
230block!1=data :REM data address for read
operation
240block!5=100 : REM number of bytes
250block?9=0 : REM sequential pointer to 0
260X%=block MOD 256
270Y%=block DIV 256
280A%=3: REM prepare to read
290CALL &FFD1 :REM do it
300CLOSE£(block?0)
310 PRINT "Close the file again"
320PRINT $data
330END
```

The status of the carry flag after the call to &FFD1 indicates whether the transfer succeeded or not. If C contains zero then the transfer was completed smoothly. If C is set to one, then the transfer failed.

The final OS call that is concerned with filing operations is OSFSC, but this is of minimal use to us when we're programming. The call is also different to the other calls that we've mentioned here because there is no direct call address for it. We have to call it at the address held in its vector. In machine code programs, this call can be made with the instruction:

```
JMP     (&21E)
```

There are other OS routines that carry out specific functions for certain filing systems. For example, OSWORD calls are used to access the Floppy Disk Controller Chip for the Disk Filing System. Also, simple filing system jobs are performed by a couple of OSBYTE calls.

That completes our look at the role of the BBC Micro's operating system in the filing operations of the machine.

DIGITAYA

This is the final instalment of our adventure game programming project series. We give here the complete listing of Digitaya, a larger game given in tandem with the Haunted Forest (see page 997).

The structure of Digitaya has many similarities to that of Haunted Forest, but the game is on a much larger scale. The original layout for Digitaya involved a map of 100 locations, on which was drawn the internal layout of a computer, complete with memory, data bus, processor and much of the other hardware found inside an average home computer. The player's task is to track down the mysterious Digitaya, who is held captive somewhere inside the machine. You must negotiate many hazards, such as the joystick port, vector table and random program bugs, using your knowledge of a computer's internal layout to rescue Digitaya.

Digitaya uses similar routines to those of the Haunted Forest to carry out the mundane functions of the game, such as moving from location to location, picking up and dropping objects and looking around. However, special routines have been added to this skeletal structure to cater for the game's special perils and pitfalls.

When playing an adventure game it is often advisable to map out your route with pencil and paper as you move around the adventure world. The primary task of the successful adventure game player is to get inside the mind of the game's creator. Recreating the map of locations used in the game is often a good step towards this goal. Digitaya is designed on a flat grid, but don't expect all adventure maps to exist in only two dimensions. Three-dimensional maps are quite feasible within the kind of program structure outlined in the project. A game could also be conceived that existed in four dimensions, the fourth dimension being time. Such a game, built on a constantly shifting map, would probably defeat all but the most skilled and experienced adventure gamers.

The listing given is for the Commodore 64, although the program will run on most machines that have a Microsoft-type BASIC. Detailed flavours are given for the BBC Micro.

Some REM statements can be removed from the listing to reduce the typing effort, but to ensure the program works correctly remove them only from the end of the program lines that include other code. Lines that contain only REMs are usually subroutine titles and their line numbers are used in the GOSUB calls. Removing such lines is likely to result in an UNDEFINED STATEMENT error.

Basic Flavours

Spectrum:

Due to the unusual way in which the Spectrum handles string arrays and variables, flavours for Digitaya are too numerous to give here. You should refer to past issues. However, to help you recode the listing, we detail the main points to look for. First, the Spectrum allows only single-character string variables. All the variable names that require conversion are given in a table on page 997.

The Spectrum also allows only fixed-length string arrays, the length of each array element being set by the DIM statement. This can cause problems if, say, a string array is dimensioned so that each element is 20 characters long. If a 15-character string is then assigned to an array element, the remaining five characters in the element are filled with space characters. We need to strip off spaces from the array element, for example, before adding it to any sentence we are constructing. The variable A$ is used to pass the array element to the subroutine and should be assigned before making the subroutine call. To illustrate its use consider this example:

Microsoft version:

```
3650 SN$="YOUR "+IV$(F,1)+" IS USELESS, THE
   FORCE INCREASES"
```

Spectrum version:

```
3650 LET S$="YOUR  ":A$=IV$(F,1):GOSUB
   8500: LETS$=S$+"  IS USELESS, THE FORCE
   INCREASES"
```

The only other problem is screen clearing. In the Commodore version this is achieved using PRINT CHR$(147). Simply replace these statements with CLS.

BBC Micro

The following lines should be substituted in the Digitaya listing:

```
1400  A$=GET$
1410  CLS
2630  REPEAT:A$=GET$:UNTIL A$="Y" OR
      A$="N"
2750  RA=RND(1)
2820  P=RND(40)+7
3890  RD=RND(1):IF RD>.65 THEN 4110:REM
      HIT
4090  P=RND(40)+7
4520  IV$(4,2)=STR$(RND(40)+7):REM
      REALLOCATE TICKET POSITION
4570  RN=RND(3)+1
5560  RA=RND(1)
```

The Digitaya Adventure

```
1030 REM **   'DIGITAYA'    **
1040 REM **   A COMPUTER    **
1050 REM ** ADVENTURE GAME **
1090 :
1100 GOSUB6090:REM READ ARRAY DATA
1110 GOSUB1290:REM STORY SO FAR
1120 P=47:REM START POINT
1130 :
1140 REM **** MAIN LOOP STARTS HERE ****
1150 :
1160 MF=0:PRINT
1170 GOSUB1440:REM DESCRIBE POSITION
1180 GOSUB1560: REM LIST EXITS
1190 GOSUB2670:REM IS P SPECIAL
1200 IF SF=1 THEN 1250:REM NEXT LOOP
1210 PRINT:INPUT"INSTRUCTIONS";IS$
1220 GOSUB1700:REM ANALYSE INSTRUCTIONS
1225 IF F=0 THEN 1210:REM INVALID INSTRUCTION
1230 GOSUB 1900:REM NORMAL INSTRUCTIONS
1240 IF VF=0 THENPRINT"I DON'T UNDERSTAND"
1250 IF MF=1 THEN 1160:REM NEW POSITION
1260 IF MF=0 THEN 1210:REM NEW INSTRUCTION
1270 END
1280 :
1290 REM **** STORY SO FAR ****
1300 SN$="WELCOME TO 'DIGITAYA'"
1310 GOSUB5880:REM FORMAT
1320 PRINT
1330 SN$="AS THE MACHINE HUMS QUIETLY. YOU LOOK AROUND."
1340 SN$=SN$+" TO THE NORTH AND SOUTH STRETCHES A WIDE HIGHWAY."
1350 SN$=SN$+" YOUR MISSION IS TO FIND THE MYSTERIOUS DIGITAYA"
1360 SN$=SN$+" AND CARRY IT TO SAFETY THROUGH ONE OF THE OUTPUT
     PORTS."
1370 SN$=SN$+".. BUT WHICH ONE ?"
1380 GOSUB5880
1390 PRINT:PRINT"PRESS A KEY TO START"
1400 GETA$:IFA$=""THEN1400
1410 PRINTCHR$(147):REM CLEAR SCREEN
1420 RETURN
1430 :
1440 REM **** DESCRIBE POSITION S/R ****
1450 SN$="YOU ARE "+LN$(P):GOSUB5880
1460 SN$="YOU SEE "
1470 REM ** SEARCH FOR OBJECT **
1480 F=0:SP$=""
1490 FOR I=1TO8
1500 IF VAL(IV$(I,2))=P THEN SN$=SN$+SP$+"A "+IV$(I,1):F=1:SP$="  "
1510 NEXTI
1520 IF F=0 THENSN$=SN$+"NO OBJECTS"
1530 GOSUB5880:REM FORMAT
1540 RETURN
1550 :
1560 REM **** LIST EXITS S/R ****
1570 EX$=EX$(P)
1580 NR=VAL(LEFT$(EX$,2))
1590 EA=VAL(MID$(EX$,3,2))
1600 SO=VAL(MID$(EX$,5,2))
1610 WE=VAL(RIGHT$(EX$,2))
1620 IF(NR OR EA OR SO OR WE)=0THEN RETURN
1630 PRINT:SN$="EXITS ARE TO THE "
1640 IF NR<>0 THEN SN$=SN$+"NORTH "
1650 IF EA<>0 THEN SN$=SN$+"EAST "
1660 IF SO<>0 THEN SN$=SN$+"SOUTH "
1670 IF WE<>0 THEN SN$=SN$+"WEST "
1675 GOSUB 5880:REM FORMAT
1690 PRINT:RETURN
1690 :
1700 REM **** ANALYSE INSTRUCTION S/R ****
1705 F=0:REM ZERO FLAG
1710 IFIS$="END" OR IS$="LIST" THEN VB$=IS$:F=1:RETURN
1720 IF IS$="LOOK" THEN VB$=IS$:F=1:RETURN
1730 :
1740 REM ** SPLIT INSTRUCTION **
1750 VB$="":NN$="":REM ZERO VERB AND NOUN
1770 LS=LEN(IS$)
1780 FOR C=1TO LS
1790 A$=MID$(IS$,C,1)
1800 IF A$=" "THEN VB$=LEFT$(IS$,C-1):NN$=RIGHT$(IS$,LS-C):F=1:C=LS
1810 NEXT
1830 IF F=0 THEN PRINT:PRINT"I NEED AT LEAST TWO WORDS"
1840 RETURN
1850 :
1900 REM **** NORMAL ACTIONS S/R ****
1910 VF=0
1920 PRINT
1930 IF VB$="GO"ORVB$="MOVE"THENVF=1:GOSUB2000
1940 IF VB$="TAKE"ORVB$="PICK"THEN VF=1:GOSUB2140
1950 IF VB$="DROP"ORVB$="PUT"THENVF=1:GOSUB2360
1960 IF VB$="LIST"ORVB$="INVENTORY"THENVF=1:GOSUB2540
1965 IF VB$="LOOK" THEN VF=1:MF=1:RETURN
1970 IF VB$="END"ORVB$="FINISH"THENVF=1:GOSUB2610
1980 RETURN
1990 :
2000 REM **** MOVE S/R ****
2010 MF=1:REM MOVE FLAG SET
2015 GOSUB8600:REM SEARCH FOR DIRECTION
2020 DR$= LEFT$(NN$,1)
2030 IFDR$<>"N"ANDDR$<>"E"ANDDR$<>"S"ANDDR$<>"W"THEN2100
2040 IF DR$="N" AND NR<>0 THEN P=NR:RETURN
2050 IF DR$="S" AND SO<>0 THEN P=SO:RETURN
2060 IF DR$="E" AND EA<>0 THEN P=EA:RETURN
2070 IF DR$="W" AND WE<>0 THEN P=WE:RETURN
2080 PRINT"YOU CANT ";IS$
2090 MF=0:RETURN
2100 REM NOUN NOT OK
2110 PRINT"WHAT IS ";NN$;" ?"
2120 MF=0:RETURN
2130 :
2140 REM **** TAKE S/R ****
2145 IV$(4,1)="TICKET TO TRI-STATE"
2150 GOSUB5730:REM IS OBJECT VALID
2160 IF F=0 THEN PRINT"THERE IS NO ";W$:RETURN
2170 REM ** IS OBJECT ALREADY TAKEN ? ****
2180 OV=F:GOSUB5830
2190 IFHF=1 THEN SN$="YOU ALREADY HAVE THE "+IV$(F,1):GOSUB5880
     :RETURN
2200 :
2210 REM ** IS OBJECT HERE **
2220 IF VAL(IV$(F,2))<>P THENSN$=IV$(F,1)+" IS NOT HERE":GOSUB5880
     :RETURN
2230 :
2240 REM ** ADD OBJECT TO LIST **
2250 AF=0:FOR J=1TO4
2260 IFIC$(J)=""THENIC$(J)=IV$(F,1):AF=1:J=4
2270 NEXTJ
2280 :
2290 REM ** CHECK FOR FULL QUOTA **
2300 IF AF=0THENPRINT"YOU ALREADY HAVE 4 OBJECTS":RETURN
2310 :
2320 SN$="YOU TAKE THE "+IV$(F,1):GOSUB5880
2330 IV$(F,2)="-1":REM DELETE POSITION ENTRY
2340 RETURN
2350 :
2360 REM ** DROP S/R **
2370 GOSUB5730:REM IS OBJECT VALID
2380 IF F=0 THEN PRINT"THERE IS NO ";NN$:RETURN
2390 :
2400 REM ** IS OBJECT HELD ? **
2410 OV=F:GOSUB5830
2420 IFHF=0THENPRINT"YOU DO NOT HAVE THE ";IV$(F,1):RETURN
2430 :
2440 REM ** DROP OBJECT **
2450 SN$="YOU DROP THE "+IV$(F,1):GOSUB5880
2460 IV$(F,2)=STR$(P):REM UPDATE OBJ POSITION
2470 :
2480 REM ** DELETE FROM HELD OBJ LIST **
2490 FORJ=1TO4
2500 IF IC$(J)=IV$(F,1)THENIC$(J)="":J=4
2510 NEXTJ
2520 RETURN
2530 :
2540 REM **** LIST INVENTORY S/R ****
2550 PRINT"OBJECTS HELD:"
2560 FORI=1TO4
2570 PRINT"   ";IC$(I)
2580 NEXTI
2590 RETURN
2600 :
2610 REM **** END GAME S/R ****
2620 PRINT:PRINT"ARE YOU SURE (Y/N) ?"
2630 GETA$:IFA$<>"Y"AND A$<>"N"THEN2630
2640 IFA$="N"THEN RETURN
2650 END
2660 :
2670 REM **** IS P SPECIAL S/R ****
2680 SF=0:REM UNSET SPECIAL FLAG
2690 IF P=37 THEN2780:REM VECTOR TABLE
2700 IF P>7 THEN 2750:REM RANDOM BUG
2710 ON P GOSUB 2850,2960,3450,3830,4180,4550,5150
2720 RETURN
2730 :
2740 REM ** RANDOM BUG **
2750 RA=RND(TI)
2760 IF RA<0.05THEN GOSUB 5420:REM BUG
2770 RETURN
2780 REM ** VECTOR TABLE **
2790 SF=1
2800 SN$="YOU ARE MOVED AT HIGH SPEED TO A NEW LOCATION":GOSUB5880
2810 FORJ=1TO1000:NEXT:REM PAUSE
2820 P=INT(RND(TI)*40+7)
2830 MF=1:RETURN
2840 :
2850 REM **** TV OUTLET S/R ****
2860 SF=1
2870 SN$="YOU HAVE ENTERED THE TV OUTLET AND THERE IS NO ESCAPE."
2880 SN$=SN$+"YOU ARE DOOMED FOREVER TO BE A TV CHAT SHOW HOST"
2890 GOSUB 5880:REM FORMAT PRINT
2900 PRINT
2910 PRINT"WELCOME TO THE SHOW......
2920 FORJ=1TO500:NEXTJ
2930 GOTO 2910
2940 END
2950 :
2960 REM **** USER PORT S/R ****
2970 SF=1
2980 SN$="ESCAPE IS AT HAND BUT THE DDR BOOKING CLERK"
2990 SN$=SN$+" BARS YOUR WAY. HE TELLS YOU THAT HE HAS BEEN
     INSTRUCTED TO"
3000 SN$=SN$+" ACCEPT INPUTS ONLY. HOWEVER HE DOES TAKE ALL MAJOR"
3010 SN$=SN$+" CREDIT CARDS."
3020 GOSUB 5880:REM FORMAT PRINT
3030 :
```

```
3040 PRINT:INPUT"INSTRUCTIONS";IS$
3050 GOSUB1700:REM ANALYSE INSTRUCTIONS
3060 GOSUB1900:REM NORMAL ACTIONS
3070 IF MF=1 THEN RETURN:REM MOVE OUT
3080 IF VF=1 THEN3040:REM NEXT INSTRUCTION
3090 IF VB$<>"GIVE" THENPRINT"I DON'T UNDERSTAND":GOTO3040
3100 :
3110 REM ** INSTRUCTION IS GIVE **
3120 GOSUB5730:REM IS OBJECT VALID
3130 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO3040:REM NEXT INSTRUCTION
3140 :
3150 REM ** IS OBJECT CREDIT CARD **
3160 IF F<>5THENPRINT"HE ONLY ACCEPTS CREDIT CARDS":GOTO3040
3170 :
3180 REM ** IS CARD CARRIED **
3190 OV=5:GOSUB5830
3200 IFHF=0THENPRINT"YOU DO NOT HAVE THE ";IV$(5,1):GOTO 3040
3210 :
3220 SN$="THE CLERK TAKES THE CARD AND SAYS 'THAT WILL DO NICELY,
     SIR'"
3230 GOSUB5880:REM FORMAT PRINT
3240 SN$="YOU ARE ALLOWED TO PASS THE BARRIER AND ENTER THE USER
     PORT"
3250 GOSUB5880:REM FORMAT PRINT
3260 :
3270 REM ** IS DIGITAYA CARRIED **
3280 OV=6:GOSUB5830
3290 IF HF=1 THEN 3380:REM SUCCESS
3300 :
3310 REM ** FAILURE **
3320 SN$="WELL DONE YOU HAVE SUCCEEDED IN ESCAPING FROM THE
     CLUTCHES"
3330 SN$=SN$+" OF THE MACHINE, BUT HAVE FAILED IN YOUR MISSION"
3340 SN$=SN$+" TO BRING BACK THE MYSTERIOUS DIGITAYA"
3350 GOSUB5880:REM FORMAT PRINT
3360 END
3370 :
3380 REM ** SUCCESS **
3390 SN$="CONGRATULATIONS, YOU HAVE SUCCEEDED IN YOUR MISSION"
3400 SN$=SN$+" TO RESCUE THE WONDEROUS DIGITAYA FROM THE"
3410 SN$=SN$+" CLUTCHES OF THE MACHINE.
3420 GOSUB5880:REM FORMAT PRINT
3430 END
3440 :
3450 REM **** CASSETTE PORT S/R ****
3460 SF=1
3470 SN$="YOU FEEL AN IRRESTIBLE FORCE PULLING YOU TOWARDS"
3480 SN$=SN$+" PERMANANENT MAGNETIC SUSPENSION"
3490 GOSUB5880:REM FORMAT
3500 NS=0:REM START COUNTING INSTRUCTIONS
3510 REM ** INSTRUCTIONS **
3520 NS=NS+1:IFNS>3THEN3770:REM SUCKED OUT
3530 PRINT:INPUT"INSTRUCTIONS";IS$
3540 GOSUB1700:REM ANALYSE INSTRUCTIONS
3550 GOSUB1900:REM NORMAL ACTIONS
3560 IFMF=1THENMF=0:PRINT"YOU CAN'T MOVE...YET":GOTO3510
3570 IFVF=1THEN3510:REM NEXT INSTRUCTION
3580 IFVB$<>"USE"THENPRINT"I DON'T UNDERSTAND":GOTO3510
3590 REM ** INSTRUCTION IS USE **
3600 GOSUB5730:REM IS OBJECT VALID
3610 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO3510
3620 :
3630 REM ** IS OBJECT BUFFER ACTIVATOR **
3640 IF F=8 THEN3680:REM OK
3650 SN$="YOUR "+IV$(F,1)+" IS USELESS, THE FORCE INCREASES"
3660 GOSUB 5880:GOTO3510:REM NEXT INSTRUCTION
3670 :
3680 OV=8:GOSUB5830:REM IS BUFF ACT HELD
3690 IFHF=0THENSN$="YOU DON'T HAVE THE "+IV$(8,1):GOSUB5880:GOTO3510
3700 :
3710 REM ** SAVED **
3720 SN$="YOU USE THE BUFFER ACTIVATOR TO COUNTER THE PULL"
3730 SN$=SN$+" INTO MAGNETIC OBLIVION. THE FORCE SUBSIDES"
3740 GOSUB5880:REM FORMAT
3750 RETURN
3760 :
3770 REM ** SUCKED OUT **
3780 SN$="THE FORCE BECOMES TOO STRONG AND YOU ARE PULLED OUT"
3790 SN$=SN$+" THROUGH THE CASSETTE PORT INTO MAGNETIC NOTHINGNESS."
3800 GOSUB 5880:REM FORMAT
3810 END
3820 :
3830 REM **** JOYSTICK PORT ****
3840 SF=1
3850 SN$="A USER WITH RED-RIMMED EYES ZAPS HIS LASER AT YOU
     REPEATEDLY."
3860 GOSUB5880:REM FORMAT
3870 :
3880 REM ** INSTRUCTIONS **
3890 RD=RND(TI):IF RD>.65THEN 4110:REM HIT
3900 PRINT:INPUT"INSTRUCTIONS";IS$
3910 GOSUB1700:GOSUB1900:REM ANALYSE INSTRUCTION
3920 IFMF=1THENMF=0:PRINT"YOU CAN'T MOVE...YET":GOTO3880
3930 IFVF=1THEN3880:REM NEXT INSTRUCTION
3940 IFVB$<>"USE"THENPRINT"I DON'T UNDERSTAND":GOTO3880
3950 GOSUB5730:REM IS OBJECT VALID
3960 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO3880:REM NEXT INSTRUCTION
3970 :
3980 REM ** IS OBJECT LASER SHIELD **
3990 IF F=3 THEN4020:REM OK
4000 SN$="YOUR "+IV$(F,1)+" IS NO USE":GOSUB5880:GOTO3880
4010 :
4020 OV=3:GOSUB5830:REM IS LASER SHIELD CARRIED
4030 IFHF=0THENSN$="YOU DO NOT HAVE THE "+IV$(3,1):GOSUB5880
     :GOTO3880
4040 :
4050 REM ** SAVED **
4060 SN$="YOU USE THE LASER SHIELD TO PROTECT YOURSELF. A BLAST
     KNOCKS"
4070 SN$=SN$+" YOU OUT OF THE JOYSTICK PORT AND BACK INTO THE
     MACHINE."
4080 GOSUB5880:REM FORMAT
4090 P=INT(RND(TI)*40+7):MF=1:RETURN
4100 :
4110 REM ** HIT **
4120 SN$="YOU ARE HIT BY THE LASER AND YOU ARE ONLY DIMLY AWARE
     THAT"
4130 SN$=SN$+" YOUR ATOMS HAVE BEEN DISTRIBUTED TO THE FOUR CORNERS"
4140 SN$=SN$+" OF THE UNIVERSE"
4150 GOSUB5880:REM FORMAT
4160 END
4170 :
4180 REM **** TRI-STATE DEVICE S/R ****
4190 SF=1
4200 SN$="A LARGE SIGN SAYS 'I/O THIS WAY' BUT AS YOU MOVE
     TOWARDS IT"
4210 SN$=SN$+" A TICKET COLLECTOR SHOUTS 'TICKETS PLEASE'
4220 GOSUB5880:REM FORMAT
4230 :
4240 REM ** INSTRUCTIONS **
4250 PRINT:INPUT"INSTRUCTIONS";IS$
4260 GOSUB1700:GOSUB1900:REM ANALYSE
4270 IFMF=1 THEN RETURN
4280 IFVF=1THEN4240:REM NEXT INSTRUCTION
4290 IFVB$<>"GIVE"ANDVB$<>"OFFER"THENPRINT"I DON'T UNDERSTAND"
     :GOTO4240
4300 REM ** INSTRUCTION IS GIVE **
4310 GOSUB5730:REM IS OBJECT VALID
4320 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO4240:REM NEXT INSTRUCTION
4330 :
4340 REM ** IS OBJECT TICKET **
4350 IF F=4 THEN4400:REM OK
4360 SN$="THE TICKET COLLECTOR SHAKES HIS HEAD AND SAYS"
4370 SN$=SN$+" 'I CANNOT ACCEPT THIS "+IV$(F,1)
4380 GOSUB5880:GOTO4240:REM NEXT INSTRUCTION
4390 :
4400 OV=4:GOSUB5830:REM IS TICKET HELD
4410 IFHF=0THENPRINT"YOU DO NOT HAVE THE TICKET":GOTO4240
4420 :
4430 REM ** OK **
4440 SN$="THE TICKET COLLECTOR ACCEPTS YOUR TICKET AND ALLOWS YOU"
4450 SN$=SN$+" TO PASS THE BARRIER."
4460 GOSUB5880:REM FORMAT
4470 REM ** DEL TICKET FROM LIST **
4480 F=0
4490 FORJ=1TO4
4500 IF IC$(J)=IV$(4,1)THENIC$(J)="":J=4
4510 NEXT J
4520 IV$(4,2)=STR$(INT(RND(TI)*40+8)):REM REALLOCATE TICKET
     POSITION
4530 P=15:MF=1:RETURN
4540 :
4550 REM **** ALU ****
4560 SF=1
4570 RN=INT(RND(TI)*3+1)
4580 IF RN=1 THEN CD$="AND"
4590 IF RN=2 THEN CD$="OR"
4600 IF RN=3 THEN CD$="NOT"
4610 SN$="MOUNTED ON THE WALL THERE ARE THREE BUTTONS MARKED"
4620 SN$=SN$+" 'AND', 'OR' AND 'NOT'. ACCESS CAN BE GAINED TO THE"
4630 SN$=SN$+" ACCUMULATOR BY PRESSING THE CORRECT BUTTON"
4640 GOSUB5880:REM FORMAT
4650 :
4660 REM ** INSTRUCTIONS **
4670 PRINT:INPUT"INSTRUCTIONS";IS$
4680 GOSUB1700:GOSUB1900:REM ANALYSE
4690 IF MF=1THEN RETURN:REM MOVE OUT
4700 IF VF=1THEN 4670:REM NEXT INSTRUCTION
4710 IFVB$="USE"OR VB$="PRESS"THEN4740
4720 PRINT"I DON'T UNDERSTAND":GOTO4670
4730 :
4740 REM ** VALID COMMAND **
4750 IF VB$="PRESS"THEN 4930
4760 REM ** COMMAND IS 'USE' **
4770 GOSUB5730:REM IS OBJECT VALID
4780 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO4670:REM NEXT INSTRUCTION
4790 :
4800 REM ** IS OBJECT CODE BOOK **
4810 IF F=7 THEN4850:REM OK
4820 SN$="YOUR "+IV$(F,1)+" IS OF NO USE":GOSUB5880
4830 GOTO4670:REM NEXT INSTRUCTION
4840 :
4850 OV=7:GOSUB5830:REM IS CODE BOOK HELD
4860 IFHF=1THEN4900:REM OK HELD
4870 SN$="YOU DO NOT HAVE THE "+IV$(7,1)
4880 GOSUB5880:GOTO4670:REM NEXT INSTRUCTION
4890 :
4900 SN$="YOU OPEN THE CODE BOOK AND FIND THE WORD '"+CD$+"'
     WRITTEN INSIDE"
4910 GOSUB5880:GOTO4670:REM NEXT INSTRUCTION
4920 :
4930 REM ** COMMAND IS PRESS **
4940 IFNN$="AND"OR NN$="OR"OR NN$="NOT"THEN4970
4950 SN$="THERE IS NO "+NN$:GOSUB5880:GOTO4670:REM NEXT INSTRUCTION
4960 :
4970 REM ** RIGHT OR WRONG **
```

IAN McKINNELL

```
4980 IFNN$=CD$ THEN GOSUB5100:RETURN
4990 GOSUB5010:RETURN
5000 :
5010 REM ** WRONG S/R **
5020 SN$="WRONG, A TRAP DOOR OPENS AND YOU FIND YOURSELF BACK"
5030 SN$=SN$+" BACK IN MAIN MEMORY"
5040 GOSUB5880:REM FORMAT
5050 IF RN=1 THEN P=39
5060 IF RN=2 THEN P=35
5070 IF RN=3 THEN P=29
5080 MF=1:RETURN
5090 :
5100 REM ** RIGHT S/R **
5110 SN$="THE GATEWAY TO THE ACCUMULATOR SWINGS OPEN AND"
5120 SN$=SN$+" YOU PASS THROUGH":GOSUB5880
5130 P=30:MF=1:RETURN
5140 :
5150 REM **** GATEWAY TO MEMORY S/R ****
5160 SF=1
5170 SN$="AN USHER GREETS YOU BUT TELLS YOU THAT YOU CANNOT BE
     ADMITTED"
5180 SN$=SN$+" UNLESS YOU GIVE AN ADDRESS":GOSUB5880
5190 REM ** INSTRUCTIONS **
5200 PRINT:INPUT"INSTRUCTIONS";IS$
5210 GOSUB1700:GOSUB1900:REM ANALYSE
5220 IF MF=1 THEN RETURN:REM MOVE OUT
5230 IF VF=1 THEN 5200:REM NEXT INSTRUCTION
5240 IF VB$<>"GIVE"THENPRINT"I DON'T UNDERSTAND":GOTO 5200
5250 :
5260 GOSUB5730:REM IS OBJECT VALID
5270 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO5200:REM NEXT INSTRUCTION
5280 :
5290 REM ** IS OBJECT ADDRESS **
5300 IF F=1 THEN5330:REM OK
5310 PRINT"HE NEEDS YOUR ADDRESS":GOTO5200
5320 :
5330 OV=1:GOSUB5830:REM IS ADDRESS CARRIED
5340 IF HF=1 THEN 5370
5350 SN$="YOU DON'T HAVE THE "+IV$(1,1):GOSUB5880:GOTO5200
5360 :
5370 REM ** OK PASS THROUGH **
5380 SN$="THE USHER LOOKS AT YOUR ADDRESS AND ALLOWS YOU TO PASS"
5390 SN$=SN$+" THROUGH":GOSUB5880
5400 P=40:MF=1:RETURN
5410 :
5420 REM **** RANDOM BUG ****
5430 SF=1
5440 SN$="A LARGE AND UGLY BUG APPEARS FROM BEHIND A CHIP"
5450 SN$=SN$+" AND LUNGES TOWARDS YOU":GOSUB5880
5460 :
5470 REM ** INSTRUCTIONS **
5480 PRINT:INPUT"INSTRUCTIONS";IS$
5490 GOSUB1700:GOSUB1900:REM ANALYSE
5500 IFMF=1THENMF=0:PRINT"YOU CAN'T MOVE...YET":GOTO5480
5510 IF VF=1THEN5480:REM NEXT INSTRUCTION
5520 IF VB$="KILL"ORVB$="FIGHT"THEN5550
5530 PRINT"I DON'T UNDERSTAND":GOTO5480
5540 :
5550 REM ** COMAND IS FIGHT/KILL **
5560 RA=RND(TI)
5570 IFRA<0.5 THEN GOSUB5600
5580 GOSUB5670:RETURN
5600 REM **** KILLED BY S/R ****
5610 SN$="YOU FIGHT WITH THE BUG. IT SHOWERS YOU WITH SPURIOUS"
5620 SN$=SN$+" ERRORS AND THEY EAT INTO YOUR BRAIN."
5630 SN$=SN$+" FINALLY YOU CAN TAKE NO MORE AND YOUR HEAD
     EXPLODES."
5640 GOSUB5880
5650 END
5660 :
5670 REM **** YOU KILL S/R ****
5680 SN$="YOU FIGHT WITH THE BUG AND THOUGH IT IS A HARD STRUGGLE"
5690 SN$=SN$+" YOU EVENTUALLY IRON IT OUT AND SURVIVE.":GOSUB5880
5700 RETURN
5710 :
5720 :
5730 REM **** VALID OJECT S/R ****
5740 NN$=NN$+" ":LN=LEN(NN$):F=0:C=1
5745 FOR K=1 TO LN
5750 IF MID$(NN$,K,1)<>" " THEN NEXTK:RETURN
5755 W$=MID$(NN$,C,K-C):C=K+1:LW=LEN(W$)
5760 FORJ=1 TO 8
5770 LI=LEN(IV$(J,1)):REM LENGTH OBJECT
5780 FORI=1TO LI
5790 IFMID$(IV$(J,1),I,LW)=W$THENF=J:I=LI:J=8:K=LN
5800 NEXT I,J,K
5810 RETURN
5820 :
5830 REM **** IS OBJECT HELD S/R ****
5840 HF=0
5850 IFIV$(OV,2)="-1"THEN HF=1
5860 RETURN
5870 :
5880 REM **** FORMAT PRINTING S/R ****
5890 LC=0: REM CHAR/LINE COUNTER
5900 OC=1: REM OLD COUNT
5910 OW$="":REM OLD WORD
5920 LL=40:REM SCREEN LINE LENGTH
5930 SN$=SN$+" DUMMY "
5940 PRINT
5950 FOR C=1 TO LEN(SN$)
5960 LC=LC+1
5970 IF MID$(SN$,C,1)=" " THENGOSUB6020
5980 NEXTC
5990 PRINT
6000 RETURN
6010 :
6020 REM **** END OF LINE CHECK S/R ****
6030 NW$=MID$(SN$,OC,C-OC+1)
6040 IF LC<LL THENPRINTOW$;:GOTO6060
6050 PRINTOW$:LC=LEN(NW$)
6060 OC=C+1:OW$=NW$
6070 RETURN
6080 :
6090 REM **** READ ARRAY DATA S/R ****
6100 REM ** READ INVENTORY **
6110 DIM IV$(8,2),IC$(4)
6120 FOR C =1TO8
6130 READ IV$(C,1),IV$(C,2)
6140 NEXT C
6150 :
6160 REM ** READ LOCATION & EXIT DATA **
6170 DIM LN$(55),EX$(55)
6180 C1=0:C2=0:REM INITIALISE CHECKSUMS
6190 FOR C=1TO54
6200 READ LN$(C),EX$(C)
6210 C1=C1+VAL(LEFT$(EX$(C),4))
6220 C2=C2+VAL(RIGHT$(EX$(C),4))
6230 NEXT C
6240 READ CA:IFCA<>C1THEN PRINT"CHECKSUM ERROR":STOP
6250 READ CB:IFCB<>C2THEN PRINT"CHECKSUM ERROR":STOP
6260 RETURN
6270 REM **** INVENTORY DATA ****
6280 DATA ADDRESS NUMBER,45,KEY,34,LASER SHIELD,25
6290 DATA TICKET TO TRI-STATE,26,DATA CREDIT CARD,28
6300 DATA DIGITAYA,30,CODE BOOK,19,BUFFER ACTIVATING DEVICE,13
6310 :
6320 REM **** LOCATION & EXIT DATA ****
6330 DATA IN THE TV OUTLET,00000000
6340 DATA IN THE USER PORT,00090100
6350 DATA IN THE CASSETTE PORT,00110000
6360 DATA IN THE JOYSTICK PORT,00130000
6370 DATA IN A TRI-STATE DEVICE,00170000
6380 DATA IN THE ARITHMETIC & LOGIC UNIT,00310016
6390 DATA AT THE GATEWAY TO MEMORY,00490000
6400 DATA ON THE I/O HIGHWAY,09000001
6410 DATA ON THE I/O HIGHWAY,10000802
6420 DATA ON THE I/O HIGHWAY,11000900
6430 DATA ON THE I/O HIGHWAY,12001003
6440 DATA ON THE I/O HIGHWAY,13531100
6450 DATA ON THE I/O HIGHWAY,14001204
6460 DATA ON THE I/O HIGHWAY,15001300
6470 DATA ON THE I/O HIGHWAY A SIGN SAYS 'S OUT H',00001400
6480 DATA IN THE DATA REGISTER,00061700
6490 DATA ON AN 8 LANE HIGHWAY,16001805
6500 DATA ON AN 8 LANE HIGHWAY,17001900
6510 DATA ON AN 8 LANE HIGHWAY,18002000
6520 DATA ON AN 8 LANE HIGHWAY,19292100
6530 DATA ON AN 8 LANE HIGHWAY,20282200
6540 DATA ON AN 8 LANE HIGHWAY,21272300
6550 DATA ON AN 8 LANE HIGHWAY,22262400
6560 DATA ON AN 8 LANE HIGHWAY,23250000
6570 DATA IN THE CHARACTER MATRIX,26360024
6580 DATA HIGH IN THE MEMORY,27352523
6590 DATA IN THE MIDDLE OF MEMORY,28342622
6600 DATA IN THE MIDDLE OF MEMORY,29332721
6610 DATA LOW IN THE MEMORY,00542820
6620 DATA IN THE ACCUMULATOR'S LAIR,00000600
6630 DATA IN A LONG CORRIDOR,00420006
6640 DATA IN AN INDEX REGISTER,31000000
6650 DATA LOW IN THE MEMORY,54403428
6660 DATA IN THE MIDDLE OF MEMORY,33393527
6670 DATA HIGH UP IN MEMORY,34383626
6680 DATA IN THE CHARACTER MATRIX,35370025
6690 DATA IN A RANDOM VECTOR TABLE,00000000
6700 DATA HIGH IN MEMORY OVERLOOKING A HIGHWAY,39003735
6710 DATA IN THE MIDDLE OF MEMORY,40003834
6720 DATA IN MEMORY - TO THE EAST IS A GATEWAY,41003933
6730 DATA LOW IN MEMORY,00004054
6740 DATA IN A CORRIDOR,00430031
6750 DATA IN A CORRIDOR,00440042
6760 DATA IN A CORRIDOR,00004543
6770 DATA IN THE ADDRESS REGISTER,00004600
6780 DATA ON A 16 LANE HIGHWAY,45004700
6790 DATA ON A 16 LANE HIGHWAY,46004800
6800 DATA ON A 16 LANE HIGHWAY,47004900
6810 DATA ON A 16 LANE HIGHWAY A LARGE GATE LOOMS TO THE WEST
     ,48005007
6820 DATA ON A 16 LANE HIGHWAY,49005100
6830 DATA ON A 16 LANE HIGHWAY,50005200
6840 DATA ON A 16 LANE HIGHWAY,51000000
6850 DATA IN A VECTOR TO MEMORY,00290012
6860 DATA LOW IN MEMORY,00413329
6870 REM ** CHECKSUM DATA **
6880 DATA 100169,103973
8599 :
8600 REM **** SEARCH FOR DIRECTION S/R ****
8610 NN$=NN$+" ":LN=LEN(NN$):C=1
8620 FORI=1 TO LN
8630 IF MID$(NN$,I,1)<>" " THEN NEXT I:RETURN
8640 W$=MID$(NN$,C,I-C):C=I+1
8650 IF W$="NORTH" OR W$="EAST" THEN NN$=W$:I=LN
8660 IF W$="SOUTH" OR W$="WEST" THEN NN$=W$:I=LN
8670 NEXT I
8680 RETURN
```

JOYSTICK JURY

Stephanie Brittain reviews three programs for the Commodore 64.

REVENGE OF THE MUTANT CAMELS

Llamasoft, £7.95

Like a million other people, I really enjoyed Attack Of The Mutant Camels. In fact I've never yet found a game by Jeff Minter that isn't exceedingly clever and innovative. But I didn't let my past experiences prejudice my assessment of this game – I was perfectly prepared to dismiss it if it really wasn't any good. Nevertheless, Minter's sequel got the better of me – it's too smashing for words.

The game gives you an amazing sense of power – using everything at your disposal to knock out shielded death-spitting camels or vaulting phone boxes while blasting away at moving CND badges intent on your destruction (some irony there, surely?). This is surrealist humour gone truly beserk: every one of the game's 32 levels (the random option means that you get a chance to see the higher levels anytime you want) is a wonder to behold.

PERCY THE POTTY PIGEON

Gremlin Graphics, £7.95

Driving in London has various hazards, including the traumas caused by the pigeon's well-known aversion to flying where walking will do, and thus sprinting out from under your wheels as it takes a leisurely stroll across the road. The eponymous pigeon featured in this game is a truly potty empyreal flapper. Percy likes nothing better than flying above the city skyline dropping bombs on cars passing on the motorway beneath – a sobering tale of carnage on the roads.

Percy, controlled by the joystick, is not totally destructive – he also likes to swoop down on the motorway (avoiding revengeful motorists) to pick up twigs, which he then takes to the nearest nest site in the roadside trees. In higher levels of play, the hand that rocks the joystick must avoid sparrows who steal the twigs and hawks who steal the pigeon.

IAN McKINNELL

CAESAR THE CAT

Mirrorsoft, £8.95

Caesar is a cute little black and white kitten whose aim is to rid the 'larder' of mice. The larder in question is, in fact, a screen crammed full with food and crockery, around which the mice scamper, eating food until Caesar catches one of them.

Points are lost if Caesar crashes into a 'wall' at the side of the screen or knocks over some crockery. You also lose points if Caesar takes too long over chasing his prey. After you've mastered the first level, you can chase faster mice on the two higher levels – and that's when the fun really begins. Mirrorsoft has certainly come up with a fresh concept, and deserves full points. There's no other game that I can compare with this for novelty – even dog lovers will like it.


abcde